ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

ENTORNO 3D PARA APRENDIZAJE COLABORATIVO DE PREPOSICIONES EN ALEMÁN A1

Raúl Gómez Sánchez

10 de diciembre de 2012

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO TÉCNICO EN INFORMÁTICA DE SISTEMAS

ENTORNO 3D PARA APRENDIZAJE COLABORATIVO DE PREPOSICIONES EN ALEMÁN A1

- Departamento: Lenguajes y Sistemas Informáticos
- Director del proyecto: Anke Berns
- Codirector del proyecto: Daniel Molina Cabrera
- Autor del proyecto: Raúl Gómez Sánchez

Cádiz, 10 de diciembre de 2012

Fdo: Raúl Gómez Sánchez

Agradecimientos

Me gustaría dedicar este texto a mi familia y en especial a mis padres, Jóse Gómez y Beatriz Sánchez, que han hecho muchos sacrificios para que puediese llegar este momento.

Agradecer a mi novia y amigos, que me han apoyado durante todo el tiempo que me ha llevado realizar el proyecto y que, por estas circunstancias, he compartido con ellos bastante menos tiempo de lo que me hubiese gustado.

También debo agradecer este proyecto a los profesores Anke Berns, Manuel Palomo y Daniel Molina, y al compañero de la Universidad Autónoma de Madrid Francisco Rodríguez; que han sido los que me han ofrecido esta oportunidad única, y que han estado pendientes del desarrollo del proyecto en todo momento.

Este programa ha sido realizado con financiación de la Actuación Avalada para la Mejora Docente "Empleo de Videojuegos para el Apoyo del Aprendizaje de Lenguas Extranjeras Aplicado al Alemán Nivel A-1" (Código AAA_012_053.XXX) de la Convocatoria de Actuaciones Avaladas para la Mejora Docente, Formación del Profesorado y Difusión de Resultados de la Unidad de Innovación Docente de la Universidad de Cádiz, Curso 2011/2012.

Personal participante:

- Anke Berns: coordinadora
- Daniel Molina Cabrera: responsable técnico
- Iván Ruíz Rube: apoyo técnico
- Juan Manuel Dodero: colaborador técnico
- Manuel Palomo Duarte: coordinador técnico
- David Camacho: colaborador técnico de la ESP de la Universidad Autónoma de Madrid
- Francisco Rodríguez: apoyo técnico de la ESP de la Universidad Autónoma de Madrid

Licencia

Notación y formato

Cuando nos refiramos a un programa en concreto, utilizaremos la notación: *emacs*.

Cuando nos refiramos a un comando, o función de un lenguaje, usaremos la notación: `quicksort`.

Cuando nos refiramos a un bloque de código, utilizaremos la notación:

```
default {
    state_entry()
    {
        llSetPrimitiveParams([PRIM_FULLBRIGHT, ALL_SIDES, FALSE]);
        llSetText("", <1.0, 1.0, 1.0>, 1.0);
        llSetPos(POSICION_INICIAL);
        llListen(CANAL_TIEMPO, "", "", "");
    }
    listen(integer channel, string name, key id, string message)
    {
        if (channel == CANAL_TIEMPO)
        {
            if (message == "start")
            {
                state actividad_empezada;
            }
        }
    }
}
```

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En los últimos años muchas instituciones educativas han basado su docencia en procesos basados en *Blended Learning*, que combina la docencia presencial con un alto porcentaje de aprendizaje autónomo y tutorización on-line. De ahí la necesidad por parte de los profesores de desarrollar nuevas herramientas de aprendizaje a fin de facilitarle a los alumnos el proceso de aprendizaje autónomo y de garantizar que adquieran los competencias esperadas. [14]

Las principales herramientas de aprendizaje autónomo on-line, o formación a través de Entornos Virtuales de Aprendizaje, que se han desarrollado en los últimos tiempos, han sido los Sistemas para la Gestión del Aprendizaje (Learning Management System o LMS). Un LMS se define como una aplicación software que se emplea para administrar, distribuir y controlar las actividades de formación no presencial. Estas aplicaciones permiten gestionar usuarios, gestionar recursos (materiales, actividades, ...), administrar el acceso, controlar y hacer seguimiento del proceso de aprendizaje, realizar evaluaciones y generar informes entre otras cosas. [13]

Entre las plataformas de LMS más usadas por profesores e instituciones académicas se encuentran (*Moodle* [1], *Sakai* [2], *Claroline* [3] o *WebCT/BlackBoard* [4]), que permiten un gran número de opciones, un diseño sencillo y un fácil uso de contenido educativo. Sin embargo, en los resultados de algunas investigaciones y estudios se ve que los estudiantes no ven los LMS como la mejor herramienta para poder aprender. Estas plataformas son consideradas "grandes repositorios de material de enseñanza", pero carecen de feedback en tiempo real y no permiten que los alumnos interactúen de forma importante. [13]

Las redes sociales son plataformas muy extendidas entre los estudiantes, pero algunas líneas de investigación muestran que éstas plataformas no tienen una gran aceptación entre los jóvenes como herramienta de aprendizaje ([10], [18]). No obstante se han intentado diseñar redes sociales creadas específicamente con fines educativos. Como ejemplo se pueden citar *GoingOn* [5] o *Edmodo* [6], que son redes sociales creadas en 2008 para que profesores y alumnos puedan compartir recursos, colaborar y aprender. A pesar de ello los mundos virtuales aportan mayor interacción y entornos 3D. ([13])

Teniendo en cuenta los aspectos antes mencionado, la codirectora del proyecto Anke Berns, junto a otros investigadores de la Universidad Autónoma de Madrid, comenzaron a explorar las posibilidades de fusionar mundos virtuales y videojuegos como herramienta para el aprendizaje autónomo on-line de los alumnos ([13] [12] [11]). La razón es que los mundos virtuales, junto con los videojuegos, representan un estimulante entorno de aprendizaje, que incrementa significativamente la motivación de los estudiantes así como su formación.

Se entiende por mundo virtual un sistema de computación usado para crear un mundo artificial donde

el usuario, a través de una representación gráfica llamada avatar, tiene la posibilidad de interactuar y manipular objetos dentro del entorno. Lo que hace particularmente atractivos a los mundos virtuales es la similitud con la apariencia del mundo real, junto con las diversas formas en la que se puede interactuar con los objetos y con los avatares. La interacción entre avatares (entre estudiantes, o entre estudiantes y profesores)se realiza en tiempo real y permite que se hagan actividades de forma cooperativa. En los mundos virtuales se pueden reproducir, simular o recrear situaciones del mundo real que introducen al usuario dentro de este mundo ([13]).
.

Los videojuegos poseen ciertas características que los hace muy atractivos, tanto a personas adultas como a jóvenes ([13]). Estas características hacen que los videojuegos tengan un gran potencial en la enseñanza y el aprendizaje ([21], [19]):

- Son inmersivos ya que los objetos y entornos están creados en 3D.
- Estimulan la cooperación y competición enfocada a la adquisición de metas.
- Proporcionan a los jugadores un feedback en tiempo real de fallos o aciertos.
- Fomentan actitudes como explorar, experimentar y asumir riesgos al resolver problemas.
- Son divertidos y muy entretenidos porque se basan en tareas orientadas.
- Soportan diferentes niveles de dificultad dependiendo de la experiencia del jugador.

Existen, sin embargo, algunos aspectos que pueden dificultar el uso de los mundos virtuales a los usuarios que usen por primera vez estos entornos 3D o que no sepan comunicarse adecuadamente en el idioma objetivo. Uno está relacionado con el hecho de que los estudiantes no sepan orientarse dentro de los grandes espacios abiertos por los que esta compuesto el mundo virtual y en los que se permite explorar los diferentes entornos donde pueden conocer, chatear y hablar con otras personas. Estas características hacen a los mundos virtuales, por un lado, poco atractivos para los principiantes, ya que normalmente carecen de las habilidades básicas del idioma necesarias para interactuar con otros estudiantes del idioma objetivo y los temas de conversación son limitados; pero por otro lado, los hace muy atractivos para los estudiantes avanzados.

En este proyecto vamos a diseñar e implantar, mediante *OpenSim*, una serie de actividades para el aprendizaje. En concreto, para que los alumnos de alemán A1 puedan practicar un aspecto que tradicionalmente les resulta difícil, el uso de preposiciones en alemán.

El proyecto que se ha desarrollado va dirigido a estudiantes de nivel A1 de alemán, por lo que es necesario garantizar que son capaces de usar adecuadamente la herramienta que le ofrecemos y que se sientan suficientemente motivados para manejarla. En ningún momento se puede permitir que los alumnos se sientan desorientados, ya que perderían interés en el mundo virtual. Para no llegar a este caso, se necesitaba crear un entorno virtual configurable y que fuese controlado totalmente por los administradores del sistema y por los profesores que hicieran uso de este mundo. Por tanto, se requería por tanto diseñar, tanto tareas específicas de acuerdo con el nivel y las necesidades de los alumnos como un sistema que controlase el seguimiento de cada las tareas de cada alumno y el progreso de aprendizaje ([13]).

Siguiendo la línea de investigación de la codirectora Anke Berns en conjunto con investigadores de la Universidad Autónoma de Madrid ([13], [12], [11]), se van a desarrollar dos actividades dirigidas al aprendizaje on-line de idioma Alemán de nivel A1. En la primera actividad (sala de entrenamiento) se

realiza un entrenamiento de vocabulario y conceptos relacionados con el uso de preposiciones en alemán, que previamente ha sido explicado en las clases presenciales y que posteriormente tendrán que aplicar en la segunda actividad. En esta tarea los alumnos deben relacionar una imagen con la palabra o el concepto adecuado, obteniendo como feedback la pronunciación en alemán de dicha palabra o dicho concepto. La segunda actividad se realiza de forma colaborativa entre dos estudiantes. La tarea consiste en ordenar los objetos de una habitación dentro de un límite de tiempo. Uno de los dos estudiantes tiene un visor con imágenes de como quedaría la habitación ordenada y, comunicándose en alemán por el chat, debe describir a su compañero en que posición van los objetos. El otro compañero dispone de un mando para mover los objetos. Al finalizar obtienen un feedback en el que se señala si las posiciones de los objetos son correctas. Ambas actividades se califican con una puntuación que los estudiantes pueden ver en el mismo momento en el que realizan la actividad, y que quedará almacenada.

Para desarrollar las actividades se ha usado el servidor 3D de mundos virtuales *OpenSim*, que se alojará en la red de ordenadores de la plataforma *VirtUAM* ([22]).

1.1. Estructura del Documento

Este documento se divide en nueve capítulos en total y algunos apéndices:

- En el capítulo 1, se muestra el trabajo de investigación realizado por la codirectora del proyecto Anke Berns, en conjunto con investigadores de la Universidad Autónoma de Madrid [13]. Se ve como está estructurada la plataforma que aloja el mundo virtual entre otros módulos ([22]). También se encuentra una breve descripción del presente proyecto.
- En el capítulo 2, se describen las plataformas elegidas o desarrolladas antes del presente proyecto, y que son usadas en este proyecto.
- En el capítulo 3, se explica cuál es el objetivo del proyecto. Consta de varias secciones en las que se muestran las características generales del proyecto.
- En el capítulo 4, se explica que modelo de ciclo de vida se ha seguido y el tiempo que se ha dedicado a cada tarea mediante un diagrama de Gantt.
- En el capítulo 5, se explica la fase de análisis que se ha realizado en el desarrollo de ambas actividades, incluyendo diagramas para facilitar la comprensión del texto.
- En el capítulo 6, se expone la fase de diseño de las actividades, es decir, se define el sistema con todo detalle.
- En el capítulo 7, se habla sobre la implementación del videojuego, las técnicas y herramientas usadas, así como las diferentes estrategias que se han seguido para realizarlo.
- En el capítulo 8, se muestran las diferentes pruebas a las que han sido sometidas las actividades para verificar que se cumplen los requisitos.
- En el capítulo 9, se exponen las conclusiones a las que he llegado después de la finalización del proyecto.
- Se incluyen distintos apéndices tales como el manual de usuario del juego, el manual de acceso al mundo virtual y el manual de instalación desde el repositorio (repositorio [15]) en el que se encuentra alojada una copia del proyecto. Al ser un producto de Software Libre se incluye el texto de la licencia en la que se basa la documentación del proyecto.

Capítulo 2

Desarrollo Previo al Proyecto

Con el fin de seleccionar el entorno de aprendizaje más apropiado, varios investigadores de la Universidad de Cádiz junto a la de la Universidad Autónoma de Madrid, comenzaron a analizar las diferentes plataformas de aprendizaje (Second Life, Active Worlds, *OpenSim*, etc) así como la posibilidad de añadir algunas herramientas adicionales. En este capítulo se profundiza en las razones por las que estos investigadores escogieron la plataforma de mundos virtuales *OpenSim* y desarrollaron la plataforma VirtUAM (Virtual Worlds at Universidad Autónoma de Madrid [22]).

2.1. VirtUAM

La plataforma VirtUAM ([22]) fue desarrollada por investigadores de la Universidad de Cádiz en colaboración con investigadores de la Universidad Autónoma de Madrid, para permitir diseñar actividades similares a videojuegos dentro del mundo virtual. Las principales diferencias de VirtUAM frente a otras propuestas similares, residen en que la plataforma desarrollada no intenta reproducir en el mundo virtual el modelo de enseñanza tradicional (clases magistrales, distribución de materiales, entrega electrónica de trabajos prácticos, ...), sino que modifica radicalmente el proceso de enseñanza, permitiendo un aprendizaje colaborativo y cooperativo entre los estudiantes. La plataforma contiene cuatro módulos que están relacionados e interconectados. Estos módulos se describen a continuación (Véase figura 2.1):

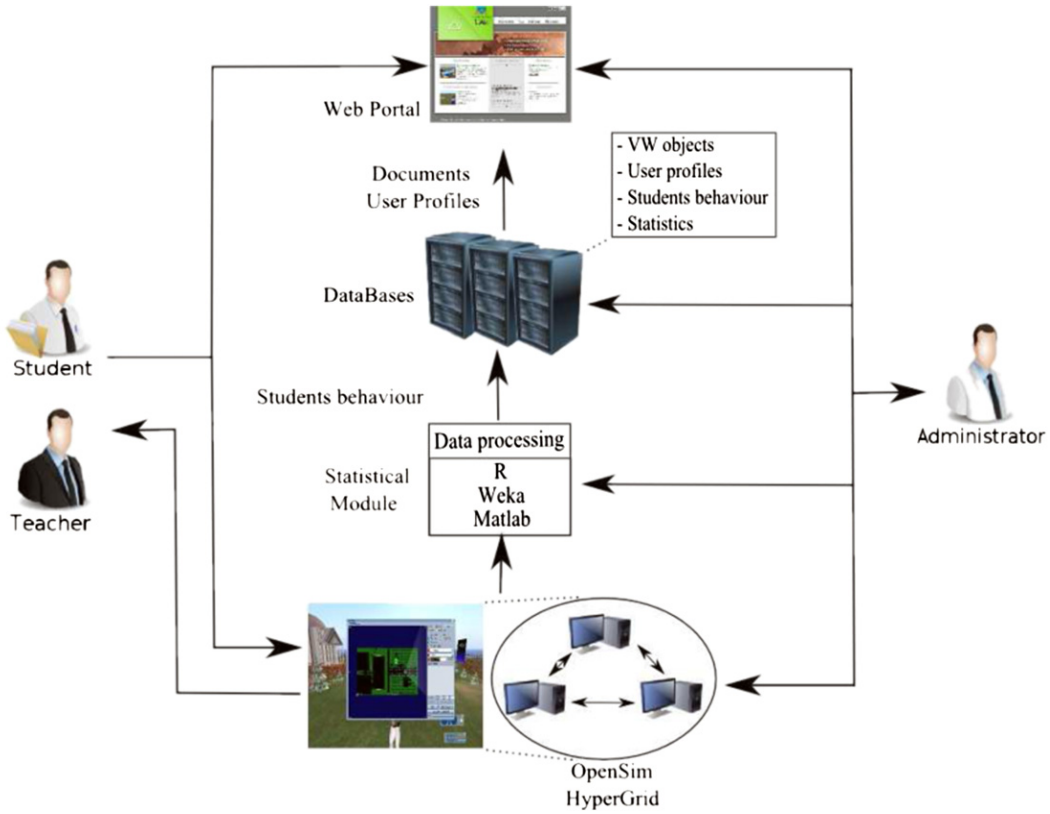


Figura 2.1: Interconexión de los diferentes módulos de VirtUAM ([13])

2.1.1. Grid

Una red de ordenadores que se encarga de alojar el mundo virtual y los juegos diseñados dentro de el. Esta red de ordenadores permite la ejecución y gestión del mundo virtual. Es necesario el uso de varios equipos con el fin de evitar problemas de rendimiento, lo que podría ocurrir, si el número de estudiantes u objetos dentro del mundo virtual aumenta.

2.1.2. Portal Web

El **Portal Web** ofrece a los usuarios (profesores, estudiantes e investigadores) acceso a diferentes documentos sobre cómo interactuar en el mundo virtual. Estos documentos se refieren a quías enfocadas a estudiantes principiantes, estudiantes avanzados, construcción dentro del mundo, tutoriales de programación, etc.

El **Portal Web** es tan importante como el **Grid**, porque permite a los estudiantes, profesores e investigadores, obtener una vista general de las acciones que pueden hacer dependiendo de su rol. Además, los profesores y los desarrolladores de aplicaciones pueden utilizar el portal web para las tareas de administración. Por ejemplo, los profesores pueden incluir un nuevo estudiante en el sistema, acceder a los perfiles de estudiantes, u obtener las estadísticas que describen el comportamiento de los estudiantes y el rendimiento del curso. Además, los diseñadores de la plataforma pueden administrar las cuentas de los maestros, analizar el rendimiento del sistema y obtener acceso a los registros almacenados en el **Sistema de Base de Datos**.

2.1.3. Sistema de Base de Datos

En el **Sistema de Base de Datos** se almacena información referente al comportamiento de los estudiantes y su interacción dentro del mundo virtual. También se almacenan registros o logs relacionados con la plataforma del mundo virtual, que pueden ser usados por profesores e investigadores para realizar análisis de datos del comportamiento de los estudiantes. El análisis de datos se puede realizar en dos niveles:

- Análisis en tiempo real, donde los datos se analizan mientras estudiantes y profesores están dentro del mundo virtual. Los resultados obtenidos permiten cambiar los juegos dinámicamente de acuerdo con las necesidades.

- Análisis al terminar el juego. En este tipo de análisis, los investigadores tratan de identificar los patrones de conducta que proporcionan mejores resultados de los estudiantes en la prueba final, que pueden ser utilizados por los profesores para diseñar juegos más eficaces.

En la base de datos se almacena la siguiente información:

- Los archivos que contienen los diferentes edificios, objetos, documentos, datos multimedia, creados en el mundo virtual. Estos archivos se utilizan para cargar el entorno virtual donde las sesiones se llevan a cabo.

- Los perfiles de estudiantes y profesores necesarios para identificar los diferentes usuarios.

- Las guías de usuario, manuales y otros documentos (es decir, sitios web, guías técnicas) para entender cómo funcionan el mundo virtual y el portal web.

- Logs sobre el comportamiento del estudiante. Estos registros se utilizan para identificar varios aspectos, por ejemplo, patrones de comportamiento.

- Las estadísticas que se utilizarán para analizar tanto el estudiante como el rendimiento del curso.

2.1.4. Módulo de Análisis Estadístico

Recibe los datos almacenados de las interacciones que se producen en el mundo virtual. Una vez que estos datos son adecuadamente asignados, se procesan mediante el uso de técnicas de minería de datos. Algunos de las conocidos herramientas de minería de datos, como Weka o R, se emplean para recuperar automáticamente los patrones de datos.

2.2. OpenSim

Figura 2.2: Logotipo de OpenSim

El proyecto *OpenSim* (*Open Simulator*) fue fundado en enero de 2007 por Darren Guard (también conocido como MW), que, como tantas otras personas, vió el potencial de un servidor de entornos virtuales 3D de código abierto que podría ser utilizado para muchas aplicaciones diferentes. Al igual que muchos otros, Darren había visto otros intentos fallidos de servidores de mundos virtuales de código abierto, a menudo debido a la enorme tarea de escribir tanto un servidor como un cliente al mismo tiempo.

Luego, en enero de 2007, el cliente de *Second Life(TM)* fue liberado como código abierto, y *libsl* (una biblioteca BSD de código abierto para crear clientes personalizados que podrían conectarse a *Second Life(TM)*), estaba llegando al punto de ser estable. Así nació la idea de *OpenSimulator*, con el objetivo inicial de probar que el concepto del servidor cliente *SL* podría conectarse y permitir algunas funciones básicas. Lo que se espera es que con el tiempo el alcance del proyecto pueda llegar a ser mucho mayor que en sus humildes comienzos. Esto ha ocurrido, con el objetivo actual de elaborar una plataforma estándar de entorno virtual que cualquier aplicación puede utilizar como marco. A pesar de que todavía mantienen la compatibilidad con el cliente de *Second Life*, han estado trabajando para apoyar a otros clientes. En el futuro se espera también soportar los protocolos y entornos que son completamente independientes de los de *Second Life*.

OpenSim es un desarrollo conjunto de código libre que tiene por objeto generar servidores de metaversos como *Second Life*. Mientras que Second Life es privado, tiene una economía real y su framework es cerrado, *OpenSim* es todo lo contrario, esta implementado en C# y puede ser ejecutado tanto en Windows sobre .NET Framework como en Linux sobre Mono Framework. Se puede conseguir el código fuente que esta amparado por una licencia tipo BSD y se puede usar en productos comerciales.

OpenSim tiene como finalidad crear entornos virtuales 3D con diferentes propósitos como educación, marketing, simulación, entretenimiento, etc. y todo en tiempo real.

Veámos a continuación una comparativa entre *Second Life* y *OpenSim*.

SECOND LIFE	OPENSIM
Gastos de 295$ al mes por cada región, más 1.000$ de configuración. Subir texturas cuesta dinero.	Gratis si se ejecuta en un servidor propio. A partir de 10$ al mes por cada región básica, e incluso 100$ al mes si es una región dentro de redes comerciales. Subir texturas es gratis.
Existen algunas tecnologías disponibles de gestión digital de contenido protegido y de derechos, pero está siendo frecuente el robo de contenido.	La protección de contenido depende completamente del propietario del grid, que puede configurarla respecto a sus necesidades. Permite realizar copias de seguridad de inventarios y regiones.
Linden Lab es propietario de todo el contenido en Second Life y los usuarios simplemente obtienen una licencia para usarlo. Linden Lab puede acceder y modificar todo (inventarios, regiones o cuentas de usuario) en cualquier momento.	Los propietarios individuales de la red determinan las políticas de uso de contenido en sus redes.
Linden Lab permite a los usuarios hacer copias de seguridad del contenido en el que ellos han creado todas las partes del objeto.	Los propietarios del grid determinan las políticas de copia de seguridad. Se pueden realizar copias de seguridad de todos los objetos, de los inventarios completos, de regiones enteras, o de todo el grid.
Second Life utiliza el motor de física Havoc.	Los propietarios de grids *OpenSim* pueden cambiar módulos de *OpenSim* para utilizar motores de física de código abierto o comerciales.
Second Life utiliza LSL, *Linden Scripting Language*.	*OpenSim* es compatible con los comandos de LSL, y añade comandos propios OSSL, *OpenSim* Scripting Language. Los usuarios también pueden definir sus propios comandos de scripting e incluirlos como módulo de *OpenSim* o crear un motor de scripting completamente nuevo (como por ejemplo Phlox de InWorldz).
Second Life utiliza el sistema de voz Vivox.	Los propietarios de grids pueden optar por instalar módulos para diferentes sistemas de voz gratuitos, incluyendo FreeSWITCH y Whisper/Mumble, o comprar una licencia comercial.
Teletransporte sólo entre las regiones en red de *Second Life*.	Teletransporte entre las redes hypergrid habilitadas mediante el uso de un Hypergate, región de enlace, o introduciendo la dirección en el campo de búsqueda en el mapa.

La razón por la que se eligió *OpenSim* es que es un software de código abierto que se puede utilizar, con los módulos de software apropiados, como una plataforma de e-Learning o LMS. Para utilizar *OpenSim* como una plataforma de e-Learning se diseñó VirtUAM. A continuación se muestran las características que posee *OpenSim* frente a otras plataformas similares:

- El espacio virtual, que puede ser construido por sus usuarios y el número de prims (formas primi-

tivas) creados en esta plataforma son ilimitados. Sin embargo, un gran número de prims, scripts o sims tiene un efecto negativo en el rendimiento del sistema.

- Por lo general, las plataformas de mundos virtuales son lugares públicos virtuales que no pueden ser completamente configurados para proporcionar un ambiente educativo exclusivamente controlada por sus usuarios, es decir, que cualquier persona podría acceder e interferir en el desarrollo de las actividades.
- La información relacionada con la conducta de los estudiantes, como estudiante-profesor y la interacción alumno-alumno o los registros de chat no pueden ser fácilmente recuperados desde plataformas externas a los mundos virtuales.
- *OpenSim* es un software de código abierto, que permite a los administradores y profesores modificar el programa cuando lo deseen. Tales modificaciones podrían tener como objetivo el almacenamiento de la conducta del jugador dentro de un sistema de base de datos o, incluso la inclusión de un mecanismo, que puede detectar problemas individuales de alumnos, a fin de centrarse específicamente en estos.

Capítulo 3

Descripción General

Este capítulo explica cuál es el objetivo del proyecto. Consta de varias secciones en las que se muestran las características generales del proyecto.

3.1. Perspectiva del Producto

Este software se ha desarrollado para fomentar el aprendizaje individual y colaborativo on-line para estudiantes de nivel A1 de alemán. En concreto el software se centra en una parte específica y que resulta difícil aprender en el idioma objetivo, el uso de las preposiciones.

Las aplicaciónes son complementarias a la enseñanza presencial y tienen como propósito afianzar los contenidos anteriormente introducidos en el aula.

3.1.1. Dependencias del producto

El presente proyecto es parte de una línea de investigación realizada por la codirectora del proyecto Anke Berns junto a otros investigadores de la Universidad Autónoma de Madrid ([13]). Este proyecto forma parte, por tanto, de un proyecto mucho mayor que se lleva desarrollando desde hace algunos años, y que continuará en un futuro.

Las actividades desarrolladas en el proyecto dependen de la plataforma VirtUAM, ya que estas actividades han sido diseñadas dentro del mundo virtual que se encuentra alojado en el módulo Grid y usan su módulo de base de datos (Sección 2.1).

3.1.2. Interfaz de usuario

El usuario interactúa a través de objetos 3D creados dentro del mundo virtual.

Para realizar las actividades desarrolladas dentro del mundo virtual, el usuario debe interactuar con los objetos por medio de su avatar. Para ello tiene que situarse cerca de estos objetos.

Se han desarrollado en total dos actividades, y el usuario para cambiar de actividad debe teletransportarse de una región a otra. Para teletransportarse el usuario tiene dos formas: usando los teletransportes

habilitados en cada una de las regiones, o usando el mapa.

La mayoría de los objetos que aparecen tanto en las imágenes de los paneles de la sala de entrenamiento, como los que están distribuidos por la habitación de la sala de ordenar la habitación, tienen colores vivos y saturados, que ayudan a centrar la atención de los alumnos en las actividades. Con esta acción conseguimos que los objetos resalten sobre el fondo compuesto por tonos suaves, y captar la atención de los alumnos sin llegar a producirles fatiga visual.

3.1.3. Interfaz software

OpenSim es multiplataforma, por lo que el servidor *OpenSim* funciona en entornos *MAC OS*, *Windows* y *Linux*. El grid de la plataforma VirtUAM en el que se aloja el mundo virtual es un entorno *Windows*.

Para poder acceder y visualizar el mundo virtual se necesita disponer de algún navegador que soporte *OpenSim*. Entre estos navegadores se encuentran *Imprudence*, *RealXtend*, *Hippo*, etc.

Como se ha mencionado en secciones anteriores el mundo virtual esta alojado en la plataforma VirtUAM (Sección 2.1), que usa como SGBD el software MySQL para almacenar registros de interacción del usuario con el entorno (chat público, chat privado, eventos que se activan en el mundo y movimiento del avatar). *OpenSim* usa internamente también MySQL como SGBD para almacenar inventarios, regiones, objetos, texturas, etc. del mundo virtual. Se puede modificar la configuración de *OpenSim* para que soporte SQLite3 o MSSQL.

Para poder comunicar el SGBD externo a *OpenSim* y el mundo virtual, creado con *OpenSim*, se utiliza un servidor capaz de ejecutar los ficheros JSP necesarios para la lectura y escritura en la Base de Datos. El software usado para esta tarea es Apache Tomcat.

3.1.4. Función del producto

El proyecto busca desarrollar actividades similares a videojuegos, que sean altamente interactivas, que fomenten el aprendizaje individual y colaborativo on-line. El objetivo en el que se va a centrar va a ser que los alumnos de nivel A1 de alemán, aprendan las preposiciones del idioma. Para ello, se han creado dos actividades dentro del mundo virtual generado con *OpenSim*, y gestionado con Virtuam.

La actividad de la sala de entrenamiento introduce nuevo vocabulario y afianza los contenidos vistos en las clases presenciales.

La actividad de ordenar la habitación hace que los alumnos utilicen el nuevo vocabulario para comunicarse entre ellos, y conseguir el objetivo final que consiste en reorganizar una habitación valiéndose de unas imágenes.

3.2. Características de los usuarios

Dentro de las regiones del mundo virtual en las que se encuentran sendas actividades, existen tres tipos de usuarios o roles:

- **Administrador** Es el responsable de instalar, mantener e interconectar con la base de datos la aplicación. Puede gestionar todos los scripts y objetos de las dos actividades y de las dos regiones en las que están alojadas. Necesita conocimientos avanzados de informática y del entorno *OpenSim*.

- **Profesor** Es el responsable de organizar las actividades entre los alumnos, explicarles el funcionamiento de las actividades, solventar aquellas dudas que puedan surgir respecto al uso del idioma objetivo, y supervisar que los alumnos sigan adecuadamente las actividades. No son necesarios conocimientos de informática, pero si un conocimiento básico del funcionamiento de *OpenSim* a nivel de usuario, de las actividades y del idioma objetivo.

- **Alumno** Esta formado por el resto de personas que participan en la actividad. A este grupo de usuarios del mundo virtual va dirigida la actividad. No necesitan conocimientos de informática, aunque si un conocimiento básico sobre el manejo de *OpenSim* a nivel de usuario.

3.3. Restricciones generales

La metodología seguida en el desarrollo del proyecto ha sido una *metodología ágil*. A lo largo del desarrollo de las dos actividades se han ido realizando reuniones en cortos periodos de tiempo con la codirectora del proyecto Anke Berns y con Francisco Rodriguez, técnico en informática de la Universidad Autónoma de Madrid. En las reuniones, ya fuesen presenciales o de forma virtual a través de *OpenSim*, se revisaban los avances hechos y se añadían nuevas funciones.

El lenguaje de programación usado es LINDEN SCRIPTING LANGUAGE (LSL) y Open Simulator Scripting Language (OSSL), que son los lenguajes de programación que proporciona *OpenSim* para añadir funcionalidades dentro del mundo virtual (aunque existen otras formas de añadir funcionalidades).

Al desarrollar cualquier actividad o aplicación dentro del mundo virtual se debe tener en cuenta que el rendimiento puede verse afectado negativamente al añadir excesivos objetos, texturas o scripts, en una misma región, o en varias regiones alojadas en la misma máquina.

3.4. Estructura de las actividades

En esta sección se ve el contenido de las salas y algunas funciones que poseen. También se muestra el aspecto final de ambas actividades terminadas.

3.4.1. Sala 1 (Juego similar a Memoria)

Esta sala se compone de varios paneles, 4 en total, y cada panel consta de 24 cuadros, un botón de inicio, un marcador de tiempo y otro de puntos. Delante de cada panel hay un sillón en el que los alumnos deben sentarse para poder jugar correctamente en el panel.

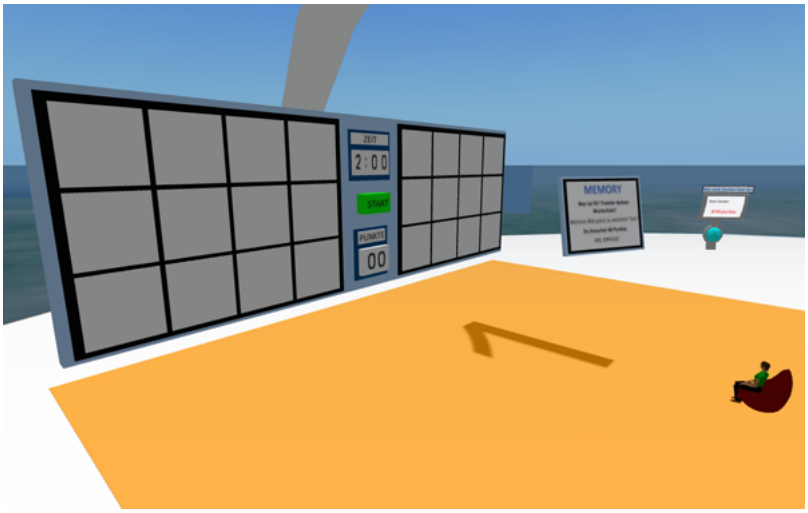


Figura 3.1: Alumno preparado para comenzar actividad

En el centro de la sala hay una pequeña cúpula con un halo morado. Esta cúpula es el teletransporte de entrada a la otra sala.

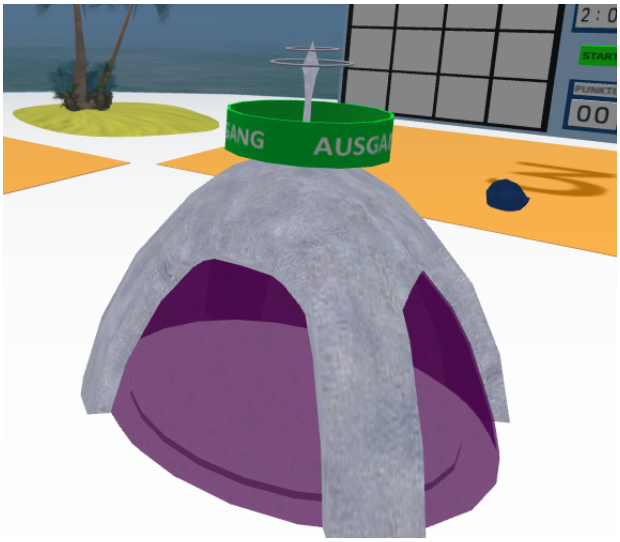


Figura 3.2: Teletransporte de la sala 1

A continuación detallo en que consiste el juego y cómo jugar.

Contenido

El número de cuadros en cada panel es 24 y cada par de cuadros tiene, una imagen de un objeto o situación entre objetos en el lado izquierdo; y un objeto escrito o la descripción de una situación entre objetos, ambos en alemán, en el lado derecho. En estos últimos cuadros también están incluidos la pronunciación del texto en alemán, que se activa al tocar y destapar el cuadro.

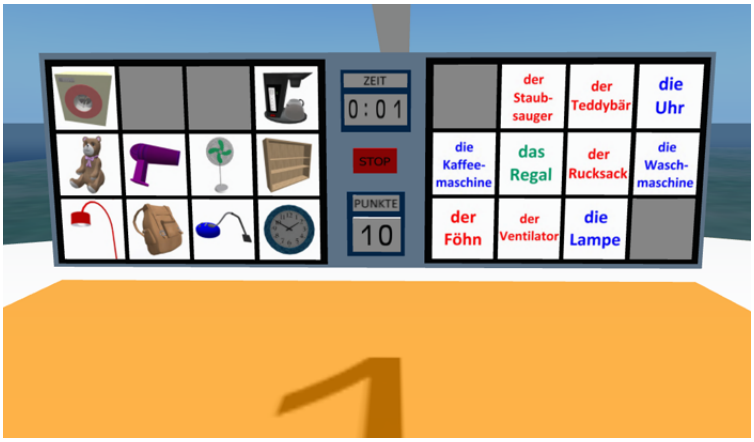


Figura 3.3: Jugando en panel 1

Mecánica del juego

Antes de comenzar el alumno debe sentarse en el sillón que hay delante de cada panel.

Para empezar el juego, se pulsa el botón de inicio y empieza la cuenta atrás de 2 minutos. Entonces el alumno debe tocar dos cuadros al azar para ponerlos boca arriba, si los dos cuadros concuerdan el jugador sumará un punto y los cuadros se quedarán boca arriba hasta el final del juego, pero si los dos cuadros no concuerdan se quedan los cuadros un rato descubiertos para que el alumno vea el contenido de cada uno y luego se colocan boca abajo nuevamente para poder levantarlas de nuevo en los próximos turnos.

Puntuación

Cada par de cuadros acertado vale un punto hasta un máximo de 12 puntos por panel, aunque no lo podrá ver hasta terminar la partida. Si al finalizar el panel el alumno ha conseguido 12 puntos sonará una música de acierto, pero si en cambio no los consigue la música que oye es de fallo.

Finalizar el juego

Hay dos formas de terminar el panel: por tiempo o pulsando el botón de parada. El tiempo que tiene cada panel es de 2 minutos en total y si se agota se hace el recuento de puntos y se muestran al alumno. El botón de parada solo lo puede tocar el alumno que comenzó la actividad, y al pulsarlo se hace el recuento de puntos y se para el cronómetro.

Puntuación Total

En uno de los huecos existentes entre los paneles hay un cartel y una pantalla en la que se puede consultar la puntuación total adquirida en todos los paneles. Para obtener la visualización de la puntuación debe pulsarse el botón, que manda una solicitud a la base de datos para consultar esta puntuación, y hay un máximo de 48 puntos.

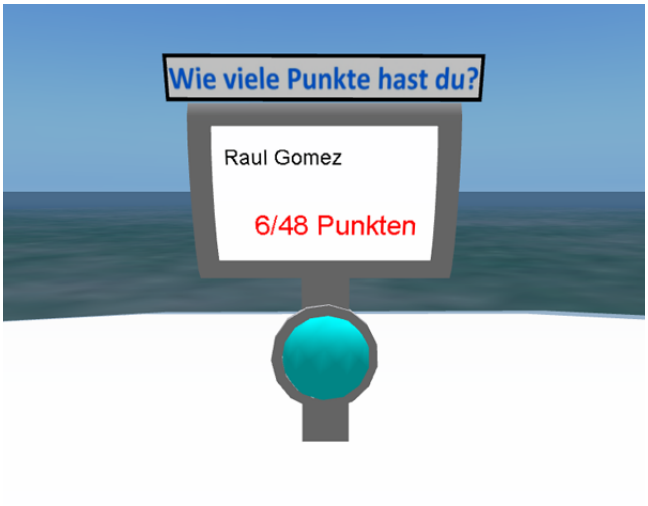


Figura 3.4: Jugando en panel 1

3.4.2. Sala 2 (Juego de ordenar la habitación)

Esta sala esta formada por los siguientes elementos una casa con dos habitaciones: una donde se va a realizar la actividad y la otra sala con una pantalla en la que poder visualizar los puntos que se han sacado en la actividad y un teletransporte a la otra sala.

Figura 3.5: Vista de parte de la casa

Contenido

En la habitación hay un botón de inicio, un marcador de puntos, un temporizador de 8 minutos y objetos cotidianos de una casa: televisor, silla, mesa, sillón . . . De todos estos objetos que tenemos repartidos por la habitación hay 29 de ellos que pueden moverse al iniciar la actividad.

Este juego se realiza por parejas y para empezar la actividad, cada avatar debe vestirse un objeto diferente. Esto significa que un avatar se vestirá un mando para poder mover los objetos de la habitación y otro avatar se viste un visor, en la que se ve la parte de la habitación a ordenador entre los dos avatares.

El mando posee 4 botones de direcciones para mover los objetos hacia delante, detrás, izquierda y derecha, en el lado izquierdo, y dos botones de colores para mover los objetos hacia arriba y abajo, en el lado derecho. El mando solo funciona al empezar la actividad.

Figura 3.6: Imagen del mando

El visor posee dos botones y una imagen en el centro. La imagen sirve para ver como quedaría la habitación si estuviese ordenada, pero al existir demasiados objetos en la sala, se han realizado 4 tomas de diferentes lugares de la casa. Los botones sirven para ir alternando estas imágenes. Si no se ve bien la imagen puede ampliarse tocando ésta. El visor se puede usar antes de comenzar la actividad.

Figura 3.7: Imágenes del visor

Mecánica del juego

Para poner en marcha el juego lo debe activar el avatar que tenga vestido el visor. Una vez activada, el avatar del visor debe dictarle, en alemán, al otro avatar las posiciónes correctas de los objetos y éste debe tocar el objeto que desea controlar y moverlo usando los botones del mando.

La comunicación debe producirse por el chat público y debe haber un supervisor de la actividad que pueda controlar que esto ocurra.

Puntuación

La puntuación solo se visualiza al finalizar la actividad y se obtiene un punto por cada objeto que esté en su lugar correcto. Los objetos no se van a ver exactamente en el lugar en el que se ha realizado la imagen, pero sí aproximádamente y es por esto que se ha incluido un margen de error en el momento de verificar la posición del objeto.

Figura 3.8: Ejemplo de finalización del juego

Cuando se finaliza la actividad también se puede ver si los objetos están situado en el lugar correcto, ya que encima de cada uno aparece un cartel que nos avisa de ello.

Finalizar el juego

Como en el juego anterior hay dos formas de acabar la actividad: tocando el botón de parar o al acabar el tiempo. El botón de parada solo lo puede activar el avatar que haya comenzado la actividad, que debe ser el mismo que tenía el visor al comenzar el juego.

Puntuación Total

En la habitación contigua a la de la actividad, hay una pantalla en la que se puede consultar la puntuación máxima adquirida en el menor tiempo posible, en esta actividad, por el avatar que consulta, que podía ser tanto el que vestía el mando como el que vestía el visor. Para obtener la visualización de la puntuación debe pulsarse el botón, que manda una solicitud a la base de datos para consultar esta puntuación, y nos muestra en la pantalla en este orden: el avatar que vestía el visor, el avatar que vestía el mando, los puntos conseguidos en esa partida y el tiempo usado en acabar dicha actividad.

Figura 3.9: Ejemplo de finalización del juego

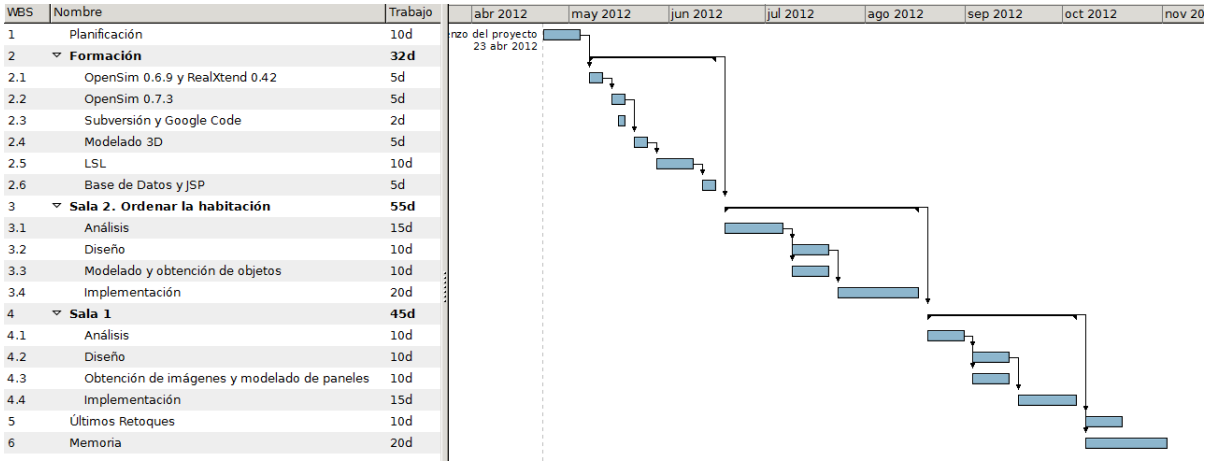
Capítulo 4

Organización Temporal

En este apartado se procede a exponer la organización temporal que se ha seguido para el desarrollo de ambas salas.

Para la realización del proyecto se ha utilizado un modelo de ciclo de vida incremental. El modelo se ha elegido debido a que no estaba demasiado claro como se iban a desarrollar las dos salas desde el principio. En adición a esto, en un primer momento, solo existía la idea de desarrollar la sala de ordenar la habitación. A medida que el desarrollo del proyecto avanzaba, la codirectora Anke Berns veía que el nivel de los alumnos era bajo y que sería más fácil, para ellos, repasar el vocabulario antes de jugar a ordenar la habitación. En esta nueva sala podrían aprender nuevo vocabulario y, practicar y recordar con el vocabulario que ya sabían, necesario para poder realizar la segunda actividad adecuadamente.

Por ello en el diagrama de Gantt se puede apreciar que la sala 2 se desarrolla en primer lugar y luego se procede al desarrollo de la sala 1, aunque hasta la parte de últimos retoques no se dan por finalizadas ninguna de las 2 salas.



WBS	Nombre	Trabajo
1	Planificación	10d
2	**Formación**	**32d**
2.1	OpenSim 0.6.9 y RealXtend 0.42	5d
2.2	OpenSim 0.7.3	5d
2.3	Subversión y Google Code	2d
2.4	Modelado 3D	5d
2.5	LSL	10d
2.6	Base de Datos y JSP	5d
3	**Sala 2. Ordenar la habitación**	**55d**
3.1	Análisis	15d
3.2	Diseño	10d
3.3	Modelado y obtención de objetos	10d
3.4	Implementación	20d
4	**Sala 1**	**45d**
4.1	Análisis	10d
4.2	Diseño	10d
4.3	Obtención de imágenes y modelado de paneles	10d
4.4	Implementación	15d
5	Últimos Retoques	10d
6	Memoria	20d

Figura 4.1: Diagrama de Gantt. Desarrollo del proyecto

En el desarrollo del proyecto se iban haciendo entregas a los tutores del proyecto para que fuesen viendo la evolución y dictando las nuevas pautas y nuevos requisitos que se deseaban introducir a los juegos, o modificar los ya realizados que no funcionasen correctamente o que no fuese exactamente lo que me habían dicho anteriormente.

Las ventajas de utilizar un este modelo iterativo incremental son las siguientes:

1. Construir un sistema pequeño es siempre menos costoso en términos de riesgo.
2. Al ir desarrollando parte de las funcionalidades, es más fácil determinar si los requisitos planeados para los niveles subsiguientes son correctos.
3. Si se comete algún error grave, sólo la última iteración necesita ser descartada.
4. Los errores de desarrollo, pueden ser arreglados antes del comienzo del próximo incremento.

Capítulo 5

Análisis del Sistema

5.1. Modelo de casos de uso

Para el desarrollo de esta documentación se utilizará la notación UML de lenguaje de modelado.

5.1.1. Modelado de la sala de entrenamiento (sala 1)

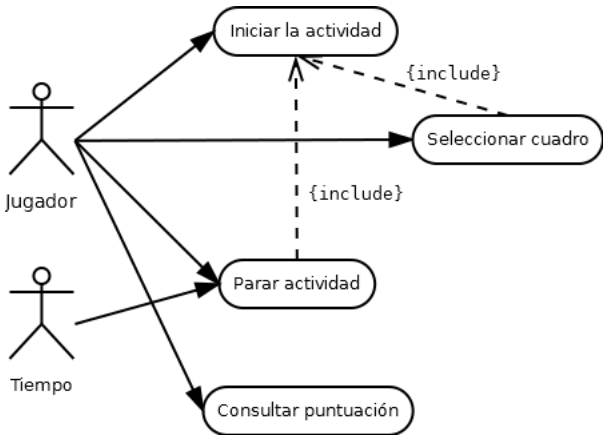


Figura 5.1: Diagrama de casos de uso de la sala de entrenamiento.

CASO DE USO: INICIAR LA ACTIVIDAD

Nivel Función.

Descripción Comenzar la actividad del panel elegido.

Actores Jugador (principal).

Precondición No hay otro jugador usando la actividad.

Postcondición Se desbloquean los cuadros, empieza a contar el tiempo y aparece el botón de STOP que solo puede usar el jugador que comenzado la actividad.

Resumen Cuando un jugador elige un panel le da al botón de START, empieza a contar el tiempo, se desbloquean los cuadros para que al tocarlos se destapen solo por el jugador que ha pulsado START.

Escenario principal El jugador comienza la actividad exitosamente

Escenario alternativo -

Escenario de error - Otro jugador quiere comenzar la actividad.

Descripción del caso de uso :

Escenario principal:

1. El jugador desea dar comienzo a la actividad del panel.
2. El jugador pulsa el botón de start.
3. El sistema registra al jugador.
4. El sistema manda la orden de desbloquear los cuadros solo para este usuario y manda otra orden para que el tiempo comience a contar.
5. El botón de START cambia por el de STOP y solo lo puede usar el usuario que ha comenzado la actividad.

Extensiones:

1a. Otro jugador desea comenzar la actividad pero ya hay otro usuario que está jugando.
   1. El sistema impide la acción del otro jugador.

CASO DE USO: SELECCIONAR CUADRO

Nivel Función

Descripción Seleccionar un cuadro del panel.

Actores Jugador (principal).

Precondición La actividad ha sido iniciado por el jugador.

Postcondición El cuadro muestra su imagen y se bloquean los cuadros restantes de su lado. Si había otro cuadro levantado y se acierta, se suma un punto, se quedan ambos cuadros bloqueados mostrando la imagen y se desbloquean todos los demás cuadros no acertados; si se falla se ocultan los dos cuadros después de unos segundos y se desbloquean todos los cuadros excepto los acertados.

Resumen Al pulsar el cuadro se muestra su imagen y se bloquean los cuadros restantes de su lado. Si había otro cuadro levantado y se acierta, se suma un punto, se quedan ambos cuadros bloqueados mostrando la imagen y se desbloquean todos los demás cuadros no acertados; si se falla se ocultan los dos cuadros después de unos segundos y se desbloquean todos los cuadros excepto los acertados.

Escenario principal El Jugador selecciona un cuadro de la parte izquierda y no hay otro cuadro en la parte derecha.

Escenario alternativo - El Jugador selecciona un cuadro de la parte derecha y no hay otro cuadro en la parte izquierda.

- El Jugador selecciona un cuadro de la parte izquierda, hay otro cuadro en la parte derecha y acierta.
- El Jugador selecciona un cuadro de la parte izquierda, hay otro cuadro en la parte derecha y falla.

- El Jugador selecciona un cuadro de la parte derecha, hay otro cuadro en la parte izquierda y acierta.
- El Jugador selecciona un cuadro de la parte derecha, hay otro cuadro en la parte izquierda y falla.

Escenario de error -El Jugador selecciona un cuadro pero no ha comenzado esa partida.

Descripción del caso de uso :

Escenario principal: Include (Iniciar la actividad)

1. El jugador pulsa un cuadro en el lado izquierdo.
2. El sistema recibe la orden, muestra la imagen y manda bloquear los demás cuadros del lado izquierdo.
3. El sistema comprueba que no hay un cuadro destapado en el lado derecho.
4. El sistema espera que el jugador pulse un cuadro del lado derecho.

Extensiones:

1a. El jugador pulsa un cuadro en el lado derecho.
    1. El sistema recibe la orden, muestra el texto, reproduce la pronunciación del texto y manda bloquear los demás cuadros del lado derecho.
    2. El sistema comprueba que no hay un cuadro destapado en el lado izquierdo.

    2a. El sistema comprueba que hay un cuadro destapado en el lado izquierdo.
        1. El sistema verifica que ambos son correctos.

        1a. El sistema verifica que ambos son incorrectos.
            1. El sistema espera unos segundos y oculta los dos cuadros.
            2. El sistema desbloquea los demás cuadros no acertados en ambos lados del panel.

        2. El sistema suma un punto al marcador y bloquea ambos cuadros con la/el imagen/-texto descubierto.
        3. El sistema desbloquea los demás cuadros no acertados en ambos lados del panel.

    3. El sistema espera que el jugador pulse un cuadro del lado izquierdo.

3a. El sistema comprueba que hay un cuadro destapado en el lado derecho.
    1. El sistema verifica que ambos son correctos.

    1a. El sistema verifica que ambos son incorrectos.
        1. El sistema espera unos segundos y oculta los dos cuadros.
        2. El sistema desbloquea los demás cuadros no acertados en ambos lados del panel.

    2. El sistema suma un punto al marcador y bloquea ambos cuadros con la/el imagen/texto descubierto.
    3. El sistema desbloquea los demás cuadros no acertados en ambos lados del panel.

CASO DE USO: PARAR ACTIVIDAD

Nivel Función.

Descripción Parar la actividad completa.

Actores Jugador (principal), Tiempo (secundario).

Precondición La actividad ha sido iniciada por el jugador.

Postcondición Se para el tiempo, se bloquean los cuadros, se guarda la puntuación y se emite un sonido de victoria o derrota según la puntuación sea 12 o menos. Pasado cierto tiempo la actividad se reinicia.

Resumen Al pulsar el botón de Stop, se bloquean todos los cuadros, se guarda la puntuación, se emite un sonido de victoria o derrota según la puntuación sea 12 o menos. Pasado cierto tiempo la actividad se reinicia y se desbloquea el botón de Start para todos los usuarios.

Escenario principal El jugador pulsa el botón de parada de la actividad, la actividad registra una puntuación de 12 y emite el sonido de victoria.

Escenario alternativo - El jugador pulsa el botón de parada de la actividad, la actividad registra una puntuación menor que 12 y emite el sonido de derrota.

- Se acaba el tiempo, se para la actividad, se registra una puntuación de 12 y emite un sonido de victoria.
- Se acaba el tiempo, se para la actividad, se registra una puntuación menor de 12 y emite un sonido de derrota.

Escenario de error - Otro jugador intenta pulsar el botón de Stop

Descripción del caso de uso :

Escenario principal:

Include (Iniciar la actividad)

1. El jugador pulsa el botón de Stop para detener la actividad.
2. El sistema manda parar el tiempo y bloquear todos los cuadros.
3. El sistema registra 12 puntos y emite un sonido de victoria.
4. Pasados 5 segundos la actividad se reinicia, se ocultan los cuadros, se pone a 0 la puntuación, el cronómetro se reinicia a 2 minutos y se desbloquea el botón de Start.

Extensiones:

1a. El cronómetro llega a 0, y manda detener la actividad.

1b. Otro jugador desea detener la actividad.

1. El sistema impide la acción del otro jugador.

3a. El sistema registra menos de 12 puntos y emite un sonido de derrota.

CASO DE USO: CONSULTAR PUNTUACIÓN

Nivel Función.

Descripción Consulta la puntuación total obtenida por el jugador en todos los paneles.

Actores Jugador (principal).

Precondición

Postcondición El jugador obtiene la puntuación obtenida en todos los paneles.

Resumen El jugador pulsa el botón de consulta la puntuación total obtenida.

Escenario principal El jugador pulsa el botón de consulta y aparece en la pantalla el nombre y la puntuación.

Escenario alternativo -

Escenario de error - El jugador no ha usado ningún panel y pulsa el botón de consulta.

Descripción del caso de uso :

Escenario principal:

1. El jugador pulsa el botón de consulta.
2. El sistema busca la puntuación del jugador.
3. El sistema muestra por pantalla el nombre del jugador y su puntuación.

Extensiones:

2a. El sistema no encuentra al jugador.

1. El sistema muestra el nombre del jugador.

5.1.2. Modelado de la sala de ordenar la habitación (sala 2)

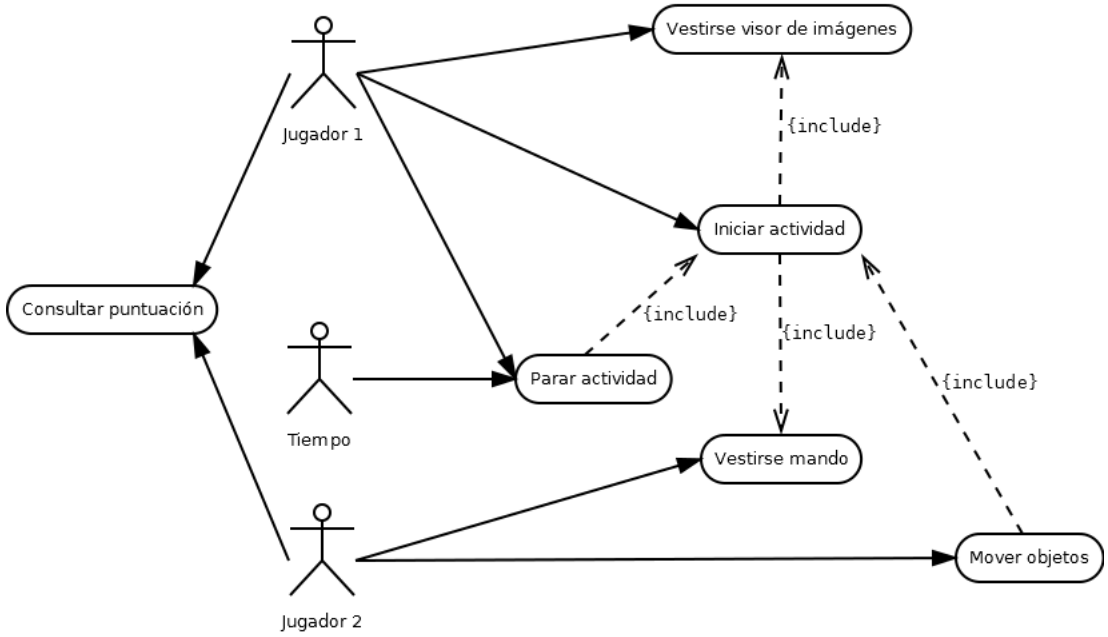


Figura 5.2: Diagrama de casos de uso de la sala de ordenar la habitación.

CASO DE USO: VESTIRSE VISOR DE IMÁGENES

Nivel Función.

Descripción Vestir el visor de imágenes al jugador 1.

Actores Jugador 1 (principal).

Precondición No hay otra pareja de jugadores usando la actividad y el jugador no tiene el mando vestido.

Postcondición El jugador visualiza el visor de imágenes en su pantalla.

Resumen El jugador 1 elige vestirse el visor de imágenes y este aparece en su pantalla para ser utilizado.

Escenario principal El jugador se viste el visor de imágenes exitósamente.

Escenario alternativo - El jugador tiene el mando vestido y se viste el visor de imágenes.

Escenario de error -

Descripción del caso de uso :

Escenario principal:

1. El jugador 1 desea vestirse el visor de imágenes.
2. El jugador 1 selecciona la opción de vestirse el visor de imágenes.
3. El sistema comprueba que no tiene vestido el mando.
4. El sistema muestra el visor de imágenes en la pantalla del jugador.

Extensiones:

3a. El sistema comprueba que tiene vestido el mando.

1. El sistema desviste el mando.

CASO DE USO: VESTIRSE MANDO

Nivel Función.

Descripción Vestir el mando al jugador 2.

Actores Jugador 2 (principal).

Precondición No hay otra pareja de jugadores usando la actividad y el jugador no tiene el visor de imágenes vestido.

Postcondición El jugador visualiza el mando en su pantalla.

Resumen El jugador 2 elige vestirse el mando y este aparece en su pantalla para ser utilizado.

Escenario principal El jugador se viste el mando exitósamente.

Escenario alternativo - El jugador tiene el visor de imágenes vestido y se viste el mando.

Escenario de error -

Descripción del caso de uso :

Escenario principal:

1. El jugador 2 desea vestirse el mando.
2. El jugador 2 selecciona la opción de vestirse el mando.
3. El sistema comprueba que no tiene vestido el visor de imágenes.
4. El sistema muestra el mando en la pantalla del jugador.

Extensiones:

3a. El sistema comprueba que tiene vestido el visor de imágenes.

1. El sistema desviste el visor de imágenes.

CASO DE USO: INICIAR LA ACTIVIDAD

Nivel Función.

Descripción Comenzar la actividad de ordenar la habitación.

Actores Jugador 1 (principal).

Precondición No hay otra pareja de jugadores usando la actividad y el jugador tiene el visor de imágenes vestido.

Postcondición Se desbloquean los objetos de la casa, empieza a contar el tiempo y aparece el botón de STOP.

Resumen Cuando el jugador 1 pulsa el botón de START, empieza a contar el tiempo, se desbloquean los objetos de la casa para que el jugador 2 pueda moverlos con el mando.

Escenario principal El jugador 1 comienza la actividad exitosamente.

Escenario alternativo -

Escenario de error - El jugador 1 no tiene vestido el visor de imágenes.
- El jugador 2 no tiene vestido el mando.
- Otro jugador quiere comenzar la actividad.

Descripción del caso de uso :

Escenario principal: Include (Vestirse el visor de imágenes), Include (Vestirse el mando)

1. El jugador 1 desea dar comienzo a la actividad de ordenar la casa.
2. El jugador 1 pulsa el botón de start.
3. El sistema registra al jugador 1, que lleva vestido el visor de imágenes, y al jugador 2, que lleva vestido el mando.
4. El sistema manda la orden de desbloquear los objetos de la casa solo para que el jugador 2 los pueda mover con el mando y envía otra orden para que el tiempo comience a contar.
5. El botón de START cambia por el de STOP y solo lo puede usar el jugador 1.

Extensiones:

1a. El jugador 1 desea comenzar a la actividad pero ya hay usuarios que están jugando.
   1. El sistema impide la acción del otro jugador.

2a. El jugador 1 no lleva vestido el visor de imágenes y pulsa el botón de start.
   1. El sistema impide la acción del jugador 1.

3a. El jugador 2 no lleva vestido el mando.
   1. El sistema no comienza la actividad.

CASO DE USO: MOVER OBJETOS

Nivel Función.

Descripción Seleccionar y mover un objeto con el mando.

Actores Jugador 2 (principal).

Precondición La pareja de jugadores ha comenzado la actividad con el jugador 2 con el mando vestido.

Postcondición El objeto seleccionado se mueve en la dirección que le indique el jugador 2 con el mando.

Resumen El jugador 2 selecciona un objeto de la casa y lo controla con los botones de dirección del mando.

Escenario principal El jugador 2 selecciona y mueve el objeto exitosamente una unidad de movimiento.

Escenario alternativo - El jugador 2 selecciona y no mueve el objeto exitosamente porque se sale de la casa.

Escenario de error - El objeto está bloqueado.

- El jugador 2 no tiene vestido el mando.
- El jugador 2 no ha seleccionado ningún objeto.
- El objeto se saldría de la casa si se moviera en la dirección indicada una unidad de movimiento.

Descripción del caso de uso :

Escenario principal: Include (Iniciar la actividad), Include (Vestirse el visor de imágenes), Include (Vestirse el mando)

1. El sistema espera que el jugador 2 seleccione un objeto.
2. El jugador 2 selecciona un objeto de la casa.
3. El sistema verifica que el objeto no está bloqueado y que el jugador 2 tiene vestido el mando.
4. El sistema espera que el jugador 2 pulse un botón de dirección del mando.
5. El jugador pulsa un botón de dirección para mover el objeto.
6. El sistema comprueba que al mover el objeto en la dirección indicada, el objeto no se sale fuera de la casa.
7. El sistema mueve el objeto en la dirección indicada una unidad de movimiento.

Extensiones:

3a. El objeto esta bloqueado.

1. El sistema impide la acción del jugador 2.

3b. El jugador 2 no tiene el mando vestido.

1. El sistema impide la acción del jugador 2.
2. El jugador 1 tiene que volver a reiniciar la partida.

5a. El jugador no ha seleccionado ningún objeto.

1. El sistema vuelve al paso 3 del escenario principal y espera a que el jugador seleccione un objeto de la casa.

6a. El sistema comprueba que al mover el objeto, éste se sale de la casa.

1. El sistema anula la orden, y vuelve al paso 4 del escenario principal.

4-7a. Repetir hasta seleccionar otro objeto, o hasta el fin de la actividad.

CASO DE USO: PARAR LA ACTIVIDAD

Nivel Función.

Descripción Parar la actividad de ordenar la habitación.

Actores Jugador 1 (principal), Tiempo (secundario).

Precondición La actividad ha sido iniciada con anterioridad.

Postcondición Se bloquean los objetos, se para el tiempo, se muestra la puntuación obtenida en la partida, se muestra si cada objeto de la casa está en su posición correcta o no, se registra la puntuación y tiempo de la actividad, y pasado un tiempo todo vuelve al estado inicial antes de comenzar el juego (marcador a 0, tiempo a 8 min, los objetos vuelven a la posición inicial sin cartel encima y se bloquean).

Resumen Cuando el jugador 1 pulsa el botón de STOP, se bloquean los objetos, se para el tiempo, se muestra la puntuación obtenida en la partida, se muestra si cada objeto de la casa está en su posición correcta o no, se registra la puntuación y tiempo de la actividad, y pasado un tiempo todo vuelve al estado inicial antes de comenzar el juego (marcador a 0, tiempo a 8 min, los objetos vuelven a la posición inicial sin cartel encima y se bloquean).

Escenario principal El jugador 1 para la actividad exitósamente.

Escenario alternativo - El tiempo llega a 0 y para la actividad exitósamente.

Escenario de error - El jugador 1 no tiene vestido el visor de imágenes.
- Otro jugador quiere parar la actividad.

Descripción del caso de uso :

Escenario principal:

Include (Iniciar actividad), Include (Vestirse el visor de imágenes)

1. El jugador 1 pulsa el botón de STOP.
2. El sistema bloquea los objetos de la casa, para el tiempo, muestra la puntuación obtenida en la partida, muestra si cada objeto de la casa está en su posición correcta o no.
3. El sistema registra la puntuación y tiempo de la actividad, junto con los nombres de los avatares que realizaron la actividad (jugador 1 y jugador 2).
4. Pasado un tiempo el sistema devuelve todo al estado inicial antes de comenzar el juego (marcador a 0, tiempo a 8 min, los objetos vuelven a la posición inicial sin cartel encima y se bloquean).

Extensiones:

1a. El tiempo llega a 0.

1b. El jugador 1 pulsa el botón pero no lleva el visor de imágenes vestido.
   1. El sistema anula la acción del jugador.

1c. Otro jugador pulsa el botón pero no lleva el visor de imágenes vestido o no es el jugador 1.
   1. El sistema anula la acción del jugador.

CASO DE USO: CONSULTAR PUNTUACIÓN

Nivel Función.

Descripción Consulta la puntuación total obtenida por el jugador en la actividad de ordenar la habitación.

Actores Jugador 1 (principal), Jugador 2 (principal).

Precondición El jugador debe haber jugado de forma exitosa, alguna vez en la actividad.

Postcondición El jugador obtiene el nombre de la pareja, la puntuación y el tiempo obtenidos en la mejor partida que haya jugado.

Resumen El jugador pulsa el botón de consulta, y obtiene el nombre de la pareja, la puntuación y el tiempo obtenidos en la mejor partida que haya jugado.

Escenario principal El jugador pulsa el botón de consulta y obtiene por pantalla el nombre de la pareja, la puntuación y el tiempo obtenidos en la mejor partida que haya jugado.

Escenario alternativo -

Escenario de error - El jugador no ha jugado y pulsa el botón de consulta.

Descripción del caso de uso :

Escenario principal:

1. El jugador pulsa el botón de consulta.
2. El sistema busca la puntuación del jugador.
3. El sistema muestra por pantalla el nombre de la pareja, la puntuación y el tiempo obtenidos en la mejor partida que haya jugado.

Extensiones:

2a. El sistema no encuentra al jugador.

1. El sistema desviste el visor de imágenes.

5.2. Modelo conceptual de datos

5.2.1. Modelado de la sala de entrenamiento (sala 1)

Para poder desarrollar la actividad de la sala de entrenamiento necesitamos las siguientes clases:

Panel: Contiene métodos y atributos para gestionar las comprobaciones de acierto o fallo, comunicarse con el marcador y cuando finaliza la actividad registrar el *UUID* del avatar y la puntuación obtenida por cada partida.

Cuadro: Posee métodos para mostrar u ocultar su contenido al jugador.

Marcador: Contiene métodos y atributos para gestionar el recuento de puntos durante la partida.

Cronómetro: Contiene métodos y atributos para gestionar el control del tiempo de la actividad.

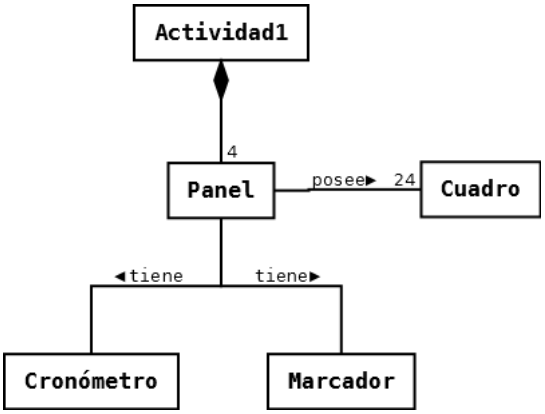


Figura 5.3: Diagrama conceptual de la sala de entrenamiento.

SUPOSICIONES

-

RESTRICCIONES TEXTUALES

- Un objeto *Panel* solo puede ser usado por un avatar al mismo tiempo. Los objetos *Cuadro*, *Cronómetro* y *Marcador* asociados a ese *Panel* también poseen la misma restricción.

5.2.2. Modelado de la sala de ordenar la habitación (sala 2)

Para poder desarrollar la actividad de la sala de ordenar la habitación necesitamos las siguientes clases:

Habitación: Contiene métodos y atributos para registrar el *UUID* de los dos avatares que controlan el objeto *Visor* y el objeto *Mando*, la puntuación obtenida y el tiempo que han usado por cada partida.

Objeto: Contiene atributos y métodos para controlar la posición en la que empieza, la posición en la que está, la posición definida como correcta y el movimiento dentro de la casa de cada objeto *Objeto*.

Marcador: Contiene métodos y atributos para gestionar el recuento de puntos durante la partida.

Cronómetro: Contiene métodos y atributos para gestionar el control del tiempo de la actividad.

Visor: Contiene atributos y métodos para visualizar cuatro imágenes de la habitación ordenada(controlar la que se desea ver y agrandarla).

Mando: Contiene atributos y métodos para comunicar la posición a la que debe moverse el objeto seleccionado de la clase *Objeto*. También controla la orientación del avatar necesaria para comunicar la órden de movimiento.

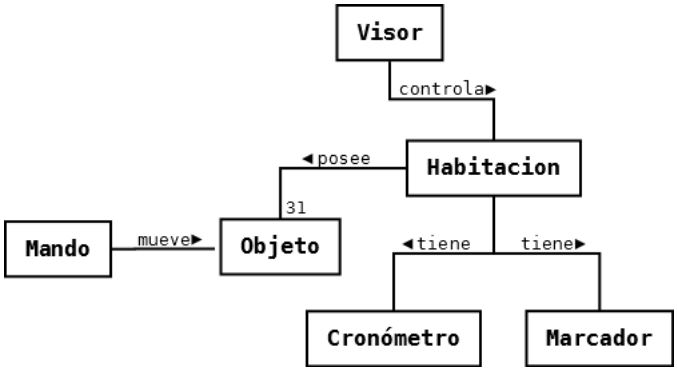


Figura 5.4: Diagrama conceptual de la sala de ordenar la habitación.

SUPOSICIONES

- Solo hay un objeto *Mando* y un objeto *Visor* en la habitación, que deben registrar a un solo avatar respectivamente, hasta el final de la actividad.

RESTRICCIONES TEXTUALES

- Un objeto *Habitación* solo puede ser usado por una pareja de avatares al mismo tiempo. Los objetos *Objeto*, *Cronómetro* y *Marcador* asociados a ese objeto *Habitación* también poseen la misma restricción.

- El objeto *Visor* solo puede registrar a un avatar por partida.

- El objeto *Mando* solo puede registrar a un avatar por partida. Este avatar puede controlar los objetos *Objeto* durante el tiempo que esté activa la actividad.

5.3. Modelo de comportamiento del sistema

Para realizar el modelo de comportamiento hemos de aclarar que se necesita conocer un poco el funcionamiento del sistema. El sistema que vamos a modelar se puede considerar orientado a objetos y en cada objeto pueden insertarse varios scripts, cada uno de los cuales internamente funciona por estados. En cada estado se pueden declarar varios eventos que se activan dependiendo de la acción que se realice sobre el objeto (estos eventos los proporciona el lenguaje). Por otra parte la forma de comunicación entre los scripts va en función de canales. Todo lo que se mande por un canal llega a todos los objetos que estén escuchando dicho canal.

El estado en el que se encuentra antes de hacer nada es el `default` y cada script debe tener uno obligatoriamente.

En este lenguaje se ha de tener en cuanta que no se crean clases. Se crean directamente los objetos con sus scripts y sus atributos deben cambiarse uno por uno.

5.3.1. Comportamiento de la sala de entrenamiento (sala 1)

CU: Inicio Actividad

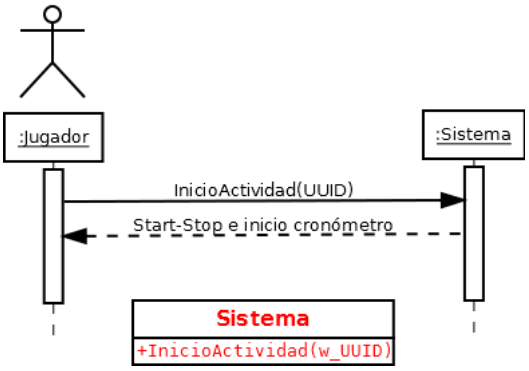


Figura 5.5: Diagrama de secuencia del caso de uso **Iniciar la Actividad** de la sala de entrenamiento.

OPERACIÓN InicioActividad (5.5)

Operación InicioActividad(w_UUID)

Responsabilidades Dar comienzo a la actividad y comunicarlo a todos los objetos implicados.

Referencias Cruzadas CU: Inicio Actividad

Precondiciones No existe actualmente un w_UUID en el objeto Start/Stop

Postcondiciones

- En el objeto Start/Stop se pasa al estado de `actividad_empezada`.
- Se inicializaron los atributos de Start/Stop (`UUID_avatar=w_UUID`).
- Se comunica a todos los objetos por CANAL_TIEMPO (Panel, a los 24 Cuadros ($Cuadro_n$), Cronómetro y Marcador) que pasen al estado de `actividad_empezada`.
- Se comunica a los objetos Panel y los 24 Cuadros por CANAL_UUID que actualicen la información derivada `/UUID_avatar`

CU: Seleccionar Cuadro

OPERACIÓN TocarCuadroIzquierdo (5.6, 5.7, 5.8)

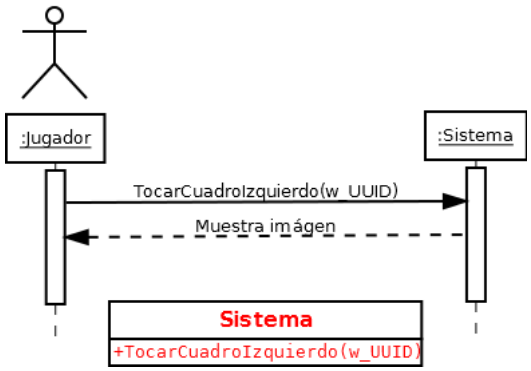


Figura 5.6: Diagrama de secuencia del caso de uso **Seleccionar Cuadro** de la sala de entrenamiento parte 1/6.

Operación TocarCuadroIzquierdo(w_UUID)

Responsabilidades Mostrar la imagen que oculta, bloquear todos los cuadros de su mismo lado del panel y comprobar si hay un cuadro en el lado derecho. Si lo hay comprobar si son pareja, bloquearlos y sumar un punto, o si no lo son ocultar ambos cuadros pasados 3 segundos. Desbloquear los cuadros de los dos lados al hacer la comprobación de si son pareja.

Referencias Cruzadas CU: Seleccionar Cuadro

Precondiciones - Los objetos Start/Stop, Panel, a los 24 Cuadros ($Cuadro_n$), Cronómetro y Marcador, están en el estado de `actividad_empezada`.

Postcondiciones - Se comprueba que `w_UUID=/UUID_avatar`.

- Se comunica por el CANAL_PANEL a todos los cuadros con `lado='i'` que pasen al estado `bloqueado`.
- Se comunica a Panel por el CANAL_PANEL los atributos `identificador` y `lado`.
- El cuadro pasa al estado `mostrado`.
- En caso de que no reciba nada por CANAL_PANEL, espera.
- En caso de que se reciba por CANAL_PANEL *acierto* pasa al estado `acertado`.
- En caso de que se reciba por CANAL_PANEL *fallo* después de 3 segundos pasa al estado `actividad_empezada`.

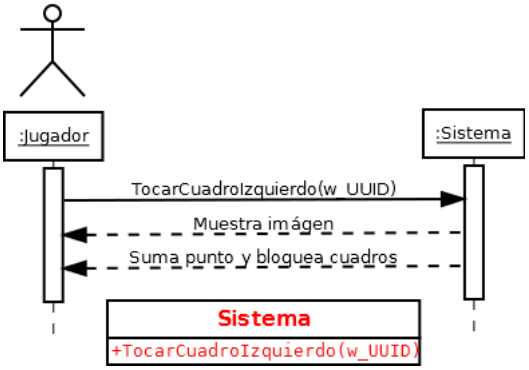


Figura 5.7: Diagrama de secuencia del caso de uso **Seleccionar Cuadro** de la sala de entrenamiento parte 2/6.

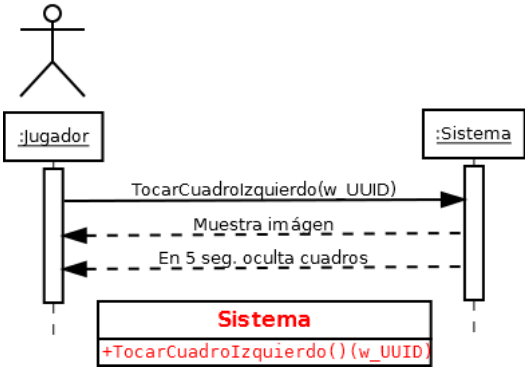


Figura 5.8: Diagrama de secuencia del caso de uso **Seleccionar Cuadro** de la sala de entrenamiento parte 3/6.

OPERACIÓN TocarCuadroDerecho (5.9, 5.10, 5.11)

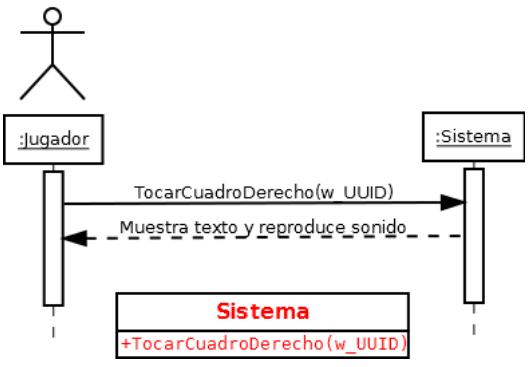


Figura 5.9: Diagrama de secuencia del caso de uso **Seleccionar Cuadro** de la sala de entrenamiento parte 4/6.

Operación TocarCuadroDerecho(w_UUID)

Responsabilidades Mostrar el texto que oculta, mandar a reproducir la pronunciación del texto, bloquear todos los cuadros de su mismo lado del panel y comprobar si hay un cuadro en el lado izquierdo. Si lo hay comprobar si son pareja, bloquearlos y sumar un punto, o si no lo son ocultar ambos cuadros pasados 3 segundos. Desbloquear los cuadros de los dos lados al hacer la comprobación de si son pareja.

Referencias Cruzadas CU: Seleccionar Cuadro

Precondiciones - Los objetos Start/Stop, Panel, a los 24 Cuadros ($Cuadro_n$), Cronómetro y Marcador, están en el estado de `actividad_empezada`.

Postcondiciones - Se comprueba que `w_UUID=/UUID_avatar`.

- Se comunica por el CANAL_PANEL a todos los cuadros con `lado='d'` que pasen al estado `bloqueado`.
- Se comunica a Panel por el CANAL_PANEL los atributos `identificador` y `lado`.
- Se comunica a Sonido por CANAL_PANEL el atributo `identificador`.
- El cuadro pasa al estado `mostrado`.
- En caso de que no reciba nada por CANAL_PANEL, espera.
- En caso de que se reciba por CANAL_PANEL *acierto* pasa al estado `acertado`.
- En caso de que se reciba por CANAL_PANEL *fallo* después de 3 segundos pasa al estado `actividad_empezada`.

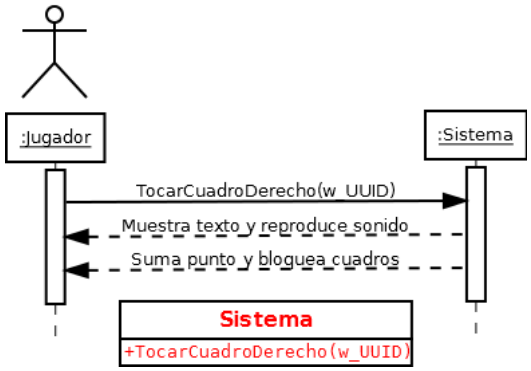


Figura 5.10: Diagrama de secuencia del caso de uso **Seleccionar Cuadro** de la sala de entrenamiento parte 5/6.

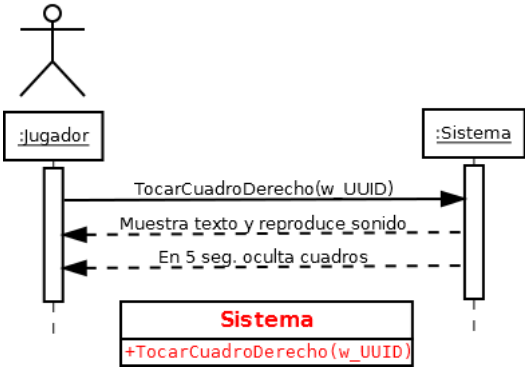


Figura 5.11: Diagrama de secuencia del caso de uso **Seleccionar Cuadro** de la sala de entrenamiento parte 6/6.

CU: Parar Actividad

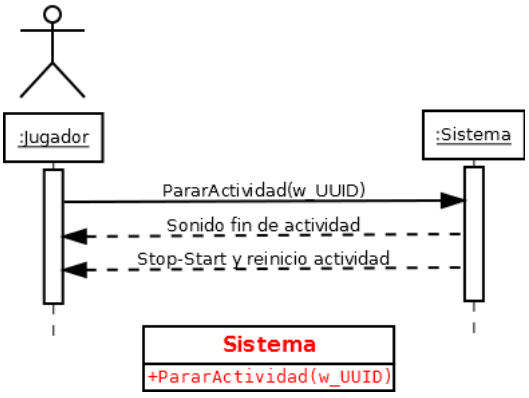


Figura 5.12: Diagrama de secuencia del sistema del caso de uso **Iniciar la Actividad** de la sala de entrenamiento.

Operación PararActividad (5.12)

Operación PararActividad(w_UUID)

Responsabilidades Detener la actividad y reiniciarla pasados unos segundos.

Referencias Cruzadas CU: Parar Actividad

Precondiciones - Los objetos Start/Stop, Panel, a los 24 Cuadros ($Cuadro_n$), Cronómetro y Marcador, están en el estado de `actividad_empezada`.

Postcondiciones - Se comprueba que `w_UUID=UUID_avatar`.

- Se mandan a la base de datos `UUID_avatar` y `puntos` desde Panel.
- Se comunica a todos los objetos por CANAL_TIEMPO (Panel, a los 24 Cuadros ($Cuadro_n$), Cronómetro y Marcador) que pasen al estado de `default`.
- Se manda desde Marcador por CANAL_PUNTOS a Sonido la puntuación, y dependiendo de este parámetro suena la melodía de acierto o fallo.

CU: Consultar Puntuación

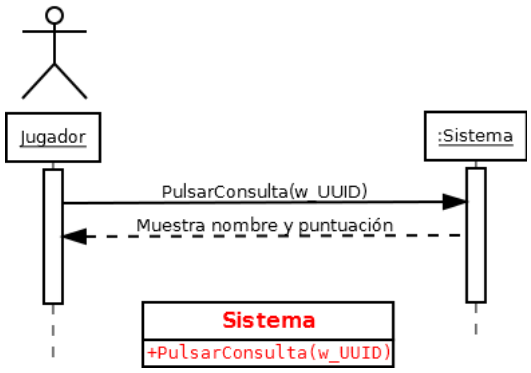


Figura 5.13: Diagrama de secuencia del sistema del caso de uso **Iniciar la Actividad** de la sala de entrenamiento.

Operación PulsarConsulta (5.13)

Operación PulsarConsulta(w_UUID)

Responsabilidades Mostrar por pantalla el nombre del avatar y la puntuación máxima obtenida en los paneles.

Referencias Cruzadas CU: Consultar Puntuación

Precondiciones - Debe existir al menos un registro de `/UUID_avatar=w_UUID`.

Postcondiciones - Se busca en todos los objetos Panel que `/UUID_avatar=w_UUID`.
- Se busca el máximo del atributo `puntos` por cada panel y se suma a `sumapuntos`.
- Se muestra por pantalla `/UUID_avatar` y `sumapuntos`.

5.3.2. Comportamiento de la sala de ordenar la habitación (sala 2)

CU: Vestirse Visor de Imágenes

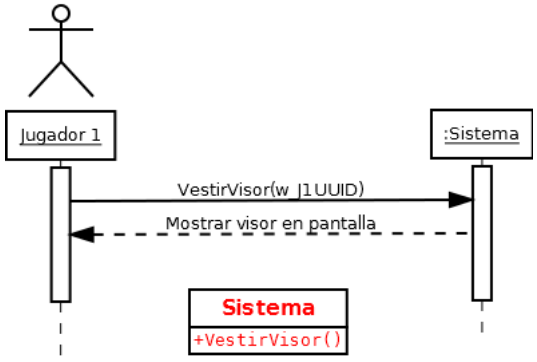


Figura 5.14: Diagrama de secuencia del sistema del caso de uso **Iniciar la Actividad** de la sala de entrenamiento.

Operación VestirVisor (5.14)

Operación VestirVisor(w_J1UUID)

Responsabilidades Mostrar por pantalla el visor de imágenes.

Referencias Cruzadas CU: Vestirse visor de imágenes.

Precondiciones - No puede haber otro avatar en la sala con el visor vestido (`UUID_avatar_visor="0"` en objeto Visor).
- El jugador no puede vestirse el visor si `UUID_avatar_visor=w_J1UUID` antes de vestirse el visor.

Postcondiciones - Se inicializa el atributo `UUID_avatar_visor=w_J1UUID`.
- Se manda a Start/Stop órden de que el visor está vestido.
- El visor se muestra en la pantalla del avatar.

CU: Vestirse Mando

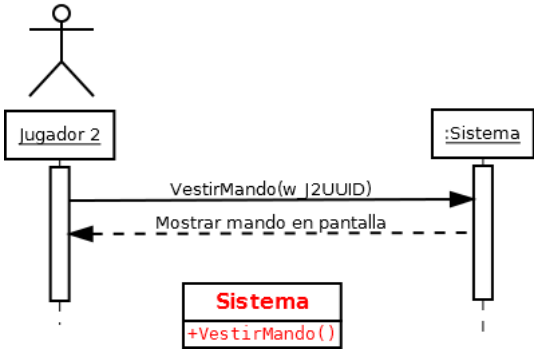


Figura 5.15: Diagrama de secuencia del sistema del caso de uso **Iniciar la Actividad** de la sala de entrenamiento.

Operación VestirMando (5.15)

Operación VestirMando(w_J2UUID)

Responsabilidades Mostrar por pantalla el mando.

Referencias Cruzadas CU: Vestirse mando.

Precondiciones - No puede haber otro avatar en la sala con el mando vestido (`UUID_avatar_mando="0"` en objeto Mando).
- El jugador no puede vestirse el mando si `UUID_avatar_mando=w_J2UUID` antes de vestirse el mando.

Postcondiciones - Se inicializa el atributo del objeto Mando `UUID_avatar_mando=w_J2UUID`.
- Se manda a Start/Stop órden de que el mando está vestido.
- El mando se muestra en la pantalla del avatar.

CU: Iniciar la Actividad

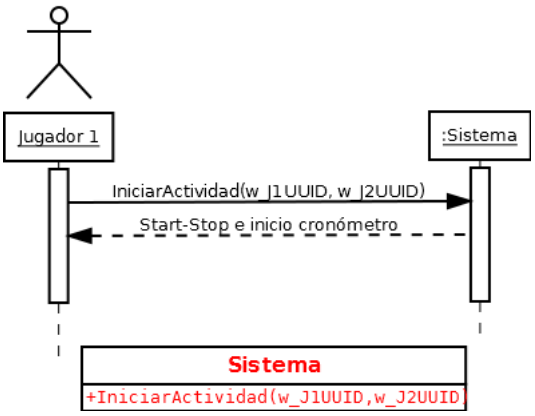


Figura 5.16: Diagrama de secuencia del sistema del caso de uso **Iniciar la Actividad** de la sala de entrenamiento.

Operación IniciarActividad (5.16)

Operación InciarActividad(w_J1UUID, w_J2UUID)

Responsabilidades Empezar la actividad de ordenar la habitación.

Referencias Cruzadas CU: Iniciar actividad.

Precondiciones - El objeto Visor tiene `UUID_avatar_visor=w_J1UUID` y el objeto Mando tiene `UUID_avatar_mando=w_J2UUID`.

Postcondiciones - En el objeto Start/Stop se pasa al estado de `actividad_empezada`.

- Se comunica a todos los objetos por CANAL_TIEMPO (Habitación, a los 31 Objetos ($Objeto_n$), Cronómetro y Marcador) que pasen al estado de `actividad_empezada`.

- Se comunica a los objetos Panel y los 31 Objetos por CANAL_UUID que actualicen la información derivada `/UUID_avatar_visor` y `/UUID_avatar_mando`

CU: Mover Objetos

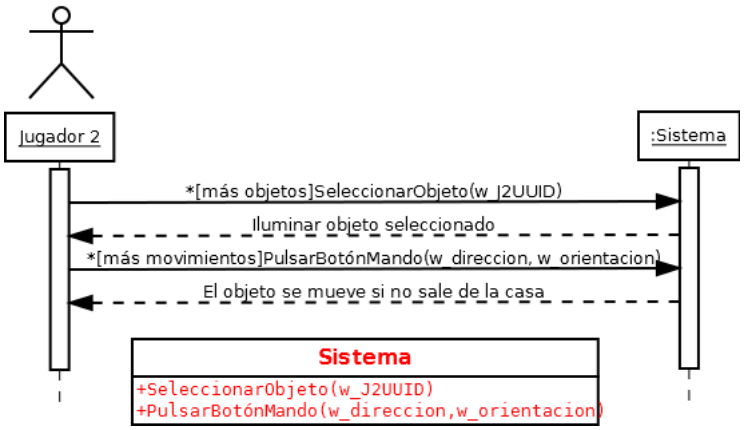


Figura 5.17: Diagrama de secuencia del sistema del caso de uso **Iniciar la Actividad** de la sala de entrenamiento.

Operación SeleccionarObjeto (5.17)

Operación SeleccionarObjeto(w_UUID, w_J2UUID)

Responsabilidades Seleccionar un Objeto de la habitación.

Referencias Cruzadas CU: Mover Objeto.

Precondiciones -

Postcondiciones - Se comprueba que el atributo derivado `/UUID_avatar_mando` coincide con `w_J2UUID`.

- El objeto Objeto pasa al estado `objeto_activo`.
- Se informa a los otros objetos Objeto que `w_J2UUID` controla a `w_UUID` por el CANAL_UUIDS y que pasen al estado `actividad_empezada`.

Operación PulsarBotonMando (5.17)

Operación

Responsabilidades PulsarBotonMando(w_direccion, w_orientacion))

Referencias Cruzadas CU: Mover Objeto

Precondiciones - El objeto previamente está seleccionado.

Postcondiciones - Se comprueba cual es la orientación del avatar `w_orientacion`.

- Se comprueba en que dirección se va a mover el objeto Objeto `w_direccion` y que se mueva dentro de los límites de la casa.
- El objeto Objeto se mueve a la nueva posición `posicion_actual`=(nueva posición).

CU: Parar Actividad

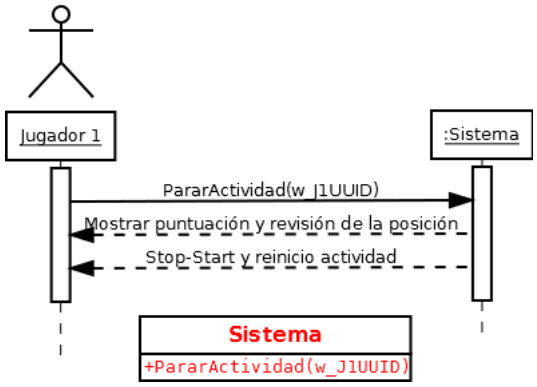


Figura 5.18: Diagrama de secuencia del sistema del caso de uso **Iniciar la Actividad** de la sala de entrenamiento.

Operación PararActividad (5.18)

Operación PararActividad(w_J1UUID)

Responsabilidades Ordena a todos los objetos que pasen al estado `reseteo` y pasado un tiempo a `default`, que se detenga el tiempo, que se muestre la puntuación y que se muestre encima de cada Objeto un cartel de si está en la posición correcta. También se mandan los datos de la partida a la base de datos.

Referencias Cruzadas CU: Parar actividad.

Precondiciones - Los objetos (Habitación, a los 31 Objetos ($Objeto_n$), Cronómetro y Marcador) deben estar en el estado `actividad_empezada` u `objeto_activo`.

Postcondiciones - Para poder parar la actividad `/UUID_avatar_visor=w_J1UUID`.

- Se mando a todos los objetos (Habitación, a los 31 Objetos ($Objeto_n$), Cronómetro y Marcador) que pasen al estado `reseteo` por el CANAL_TIEMPO.
- Los objetos Objeto mandan por CANAL_PUNTUACION a Marcador si `posicion_actual=posicion_`
- El objeto Marcador suma todos los puntos y actualiza el atributo `puntos`, muestra la puntuación y manda por CANAL_PUNTUACION el atributo `puntos` a Habitación.
- El objeto Cronómetro actualiza el atributo `tiempo_trancurrido` y lo manda por el CANAL_TIEMPO a Habitación.
- Habitación registra los atributos (`/UUID_avatar_visor, /UUID_avatar_mando, /puntos y /tiempo_transcurrido`) en la base de datos.
- Pasados 60 segundos en el estado `reseteo` se pasa al estado `default`.

CU: Consultar Puntuación

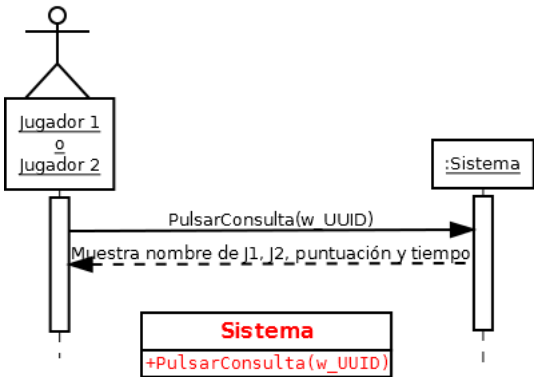


Figura 5.19: Diagrama de secuencia del sistema del caso de uso **Iniciar la Actividad** de la sala de entrenamiento.

Operación PulsarConsulta (5.19)

Operación PulsarConsulta(w_UUID)

Responsabilidades Mostrar por pantalla el nombre del avatar con `/UUID_avatar_visor`, el nombre del avatar con `/UUID_avatar_mando`, `/tiempo_transcurrido` y la puntuación máxima obtenida.

Referencias Cruzadas CU: Consultar Puntuación

Precondiciones - Debe existir al menos un registro de `/UUID_avatar_mando=w_UUID` o `/UUID_avatar_vi`

Postcondiciones - Se busca en el objeto Habitación que `/UUID_avatar_visor=w_UUID` o `/UUID_avatar_mando=w_UUID`.

- Se busca el máximo del atributo `puntos` del objeto Habitación.
- Se muestra por pantalla el nombre del avatar con `/UUID_avatar_visor`, el nombre del avatar con `/UUID_avatar_mando`, `/tiempo_transcurrido` y la puntuación máxima obtenida.

Capítulo 6

Diseño del Sistema

En este capítulo se procede a definir el sistema con todo detalle. Para ello nos ayudaremos de diagramas UML.

6.1. Diagrama de clases

A continuación se muestran dos apartados, en cada uno de los cuales se muestra una imagen del diagrama de clases que se realiza para cada actividad.

6.1.1. Diagrama de clases de la actividad 1

En la siguiente figura (6.1) podemos observar el diagrama de clases de la actividad 1 o actividad de entrenamiento. Este diagrama se ha realizado en UML.

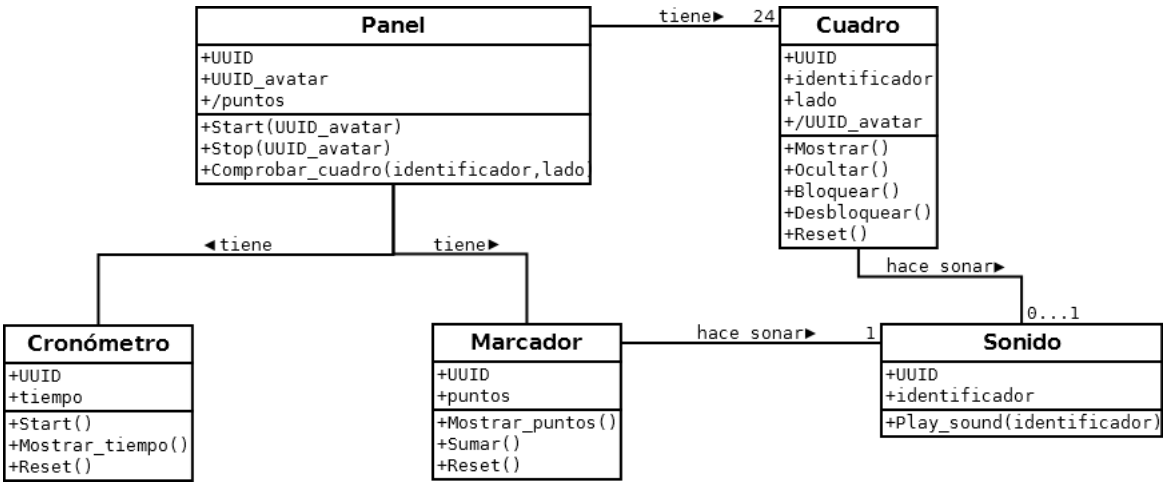


Figura 6.1: Diagrama de clases de la actividad de entrenamiento o actividad 1.

RESTRICCIONES DE CLAVE EXTERNA

Cada objeto, script, avatar, etc que se crea en el mundo virtual obtiene automáticamente una *UUID* (identificador universal único).

INFORMACIÓN DERIVADA

CLASE PANEL

***/puntos*:** puntuación que el alumno obtiene al final de la actividad y que quedará registrada en el sistema. La información se deriva de la clase **Marcador**.

CLASE CUADRO

***/UUID_avatar*:** identificador que servirá para que el cuadro solo pueda voltearse por el avatar que comenzó la actividad. La información se deriva de la clase **Panel**.

6.1.2. Clases de la actividad 1

PANEL

Contiene métodos y atributos para gestionar las comprobaciones de acierto o fallo, comunicarse con el marcador y cuando finaliza la actividad registrar el UUID del avatar en */UUID_avatar*y la puntuación obtenida por cada partida.

Start(UUID_avatar): Este método es llamado al pulsar el botón de START. El UUID del avatar se almacena en *UUID_avatar*. Es el encargado de dar comienzo a la actividad llamando a los métodos **Start()** de *Cronómetro* y **Desbloquear()** de todos los *Cuadro*.

Stop(UUID_avatar_visor) Este método es llamado al pulsar el botón de STOP o al finalizar el tiempo. Si se activa al tocar el botón de STOP se debe tener en cuenta que solo lo puede hacer el avatar que tenga el mismo UUID que el atributo *UUID_avatar*. Es el encargado de parar la actividad llamando a los métodos **Reset()** de *Marcador*, de *Cronómetro* y de todos los *Cuadro*. Llama al método **Play_sound(identificador)** con el identificador apropiado dependiendo de la puntuación */puntos*. También se encarga de registrar en la base de datos *UUID_avatar* y */puntos*.

Comprobar_cuadro(identificador,lado) Se encarga de comprobar si las parejas de cuadros son correctas en caso de que el jugador haya tocado dos cuadros. Cuando la pareja de cuadros es correcta se llama a **Bloquear()** de los *Cuadros* acertados, **Desbloquear()** de los demás *Cuadros* y **Sumar()** de *Marcador*. Cuando la pareja de cuadros es incorrecta se llama a **Ocultar()** de los *Cuadros* fallados y **Desbloquear()** de los demás *Cuadro*. En caso de que el jugador solo haya seleccionado un solo *Cuadro*, se almacena en una variable temporal el *identificador* y el *lado*, llama a **Bloquear()** de todos los *Cuadro* con *lado* = *lado* y espera a que el jugador seleccione otro *Cuadro*.

Internamente para comprobar si dos *Cuadro* son pareja, el método **Comprobar_cuadro(identificador,lado)** compara que los *identificador* de las parejas sean iguales y si lo son, compara *lado* que los lados no lo sean.

CUADRO

Posee métodos para mostrar u ocultar su contenido al jugador y comunicarlo a *Panel*. El atributo *identificador* tiene como finalidad crear las parejas y por tanto hay dos *Cuadro* con *identificador* igual por cada panel. Para diferenciar los *Cuadro* con el mismo identificador, se usa el atributo *lado*.

Mostrar() Comprueba que el avatar que ha tocado el *Cuadro* es el mismo que inicio la actividad comparando su UUID con */UUID_avatar*. Si lo es, muestra su contenido al jugador. En caso de que seleccione un cuadro de la izquierda del panel muestra una imagen. En caso de que seleccione

un cuadro de la derecha del panel muestra un texto y llama al método **Play_sound(identificador)** con el audio asociado a ese *Cuadro*. En ambos caso se llama al método **Bloquear** de todos los *Cuadro* con *lado* = *lado*.

Ocultar() Oculta el contenido del *Cuadro*.

Bloquear() Bloquea el *Cuadro* en el estado de mostrado u oculto en el que se encuentre en ese momento.

Desbloquear() Desbloquea el *Cuadro* en el estado oculto.

Reset() Pasado un tiempo se llama al método **Ocultar()** del *Cuadro* y luego a **Bloquear()**.

MARCADOR

Contiene métodos y atributos para gestionar el recuento de puntos durante la partida.

Mostrar_puntos() Muestra los puntos guardados en el atributo *puntos*.

Sumar() Suma un punto al atributo *puntos*.

Reset() Pasado un tiempo pone el atributo *puntos* a cero.

CRONÓMETRO

Contiene métodos y atributos para gestionar el control del tiempo de la actividad.
El atributo *tiempo* tiene por defecto 120 segundos. El atributo *tiempo_transcurrido* se actualiza al terminar la actividad.

Start() Se inicia la cuenta atrás con una variable que contiene el valor del atributo *tiempo*.

Mostrar_tiempo() Muestra el cambio de tiempo.

Reset() Para la cuenta atrás, asignando el tiempo que ha pasado al atributo *tiempo_transcurrido*. Pasado un tiempo se actualiza la variable con el atributo *tiempo*.

SONIDO

Contiene métodos y atributos para gestionar el control de audios y sonidos de la actividad.

El atributo *identificador* contiene los identificadores de los diferentes audios y sonidos que actúan en el juego.

Play_sound(identificador) Comprueba si el identificador es válido, y en caso de que lo sea, reproduce el audio o sonido.

6.1.3. Diagrama de clases de la actividad 2

En la siguiente figura (6.2) podemos observar el diagrama de clases de la actividad de ordenar la habitación o actividad 2. Este diagrama se ha realizado en UML.

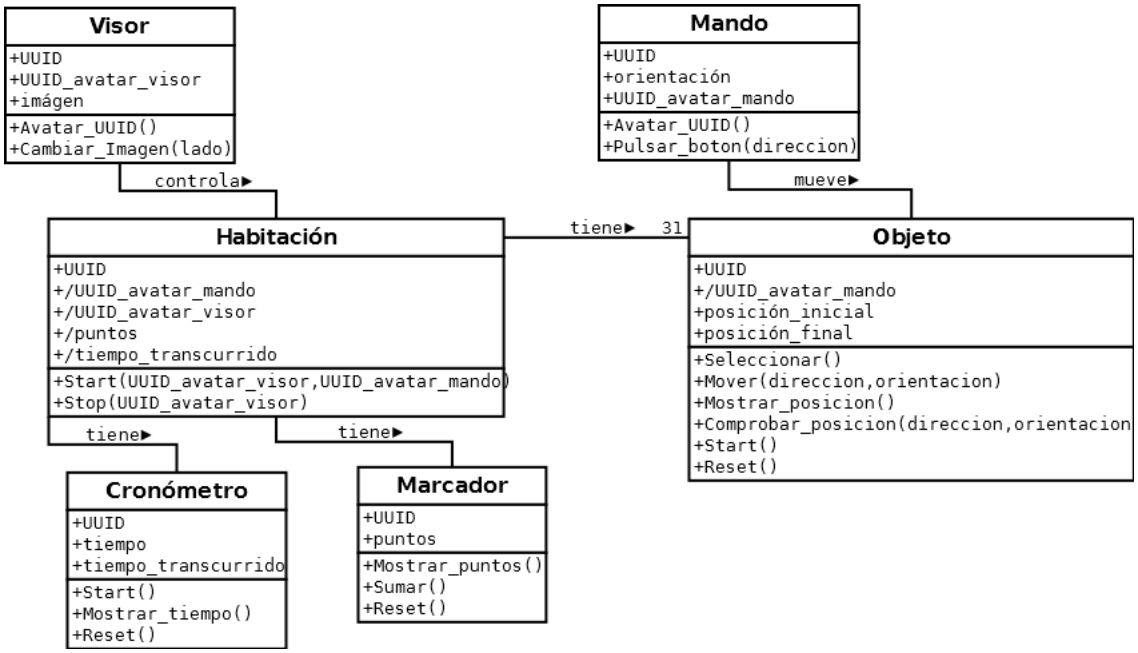


Figura 6.2: Diagrama de clases de la actividad de ordenar la habitación o actividad 2.

RESTRICCIONES DE CLAVE EXTERNA

Cada objeto, script, avatar, etc que se crea en el mundo virtual obtiene automáticamente una UUID (identificador universal único).

INFORMACIÓN DERIVADA

CLASE HABITACIÓN

***/UUID_avatar_mando*:** identificador del avatar que lleva vestido el mando que quedará registrado en el sistema. La información se deriva de la clase **Mando**.

***/UUID_avatar_visor*:** identificador del avatar que lleva vestido el visor que quedará registrado en el sistema. La información se deriva de la clase **Visor**.

***/puntos*:** puntuación que el avatar obtiene al final de la actividad y que quedará registrada en el sistema. La información se deriva de la clase **Marcador**.

***/tiempo_trancurrido*:** indica el tiempo que ha transcurrido desde que se pulso el botón de inicio hasta que se pulso el de parada, o termino el tiempo. La información se deriva de la clase **Cronómetro**

CLASE OBJETO

***/UUID_avatar_mando*:** identificador del avatar que lleva vestido el mando que servirá para desbloquear exclusivamente a este avatar los objetos durante el tiempo que dure la partida. La información se deriva de la clase **Mando**.

6.1.4. Clases de la actividad 2

HABITACIÓN

Contiene métodos y atributos para registrar el *UUID* de los dos avatares que controlan el objeto *Visor* y el objeto *Mando*, la puntuación obtenida y el tiempo que han usado por cada partida.

Los atributos que contiene son derivados de las otras clases, pero los necesita porque va a ser la clase encargada de registrar los datos (en la base de datos) al finalizar la actividad.

Sus métodos son los siguientes:

Start(UUID_avatar_visor, UUID_avatar_mando): Este método es llamado al pulsar el botón de START. Solo se puede activar por el avatar que lleve vestido el mando, y si existe un UUID de avatar en *Visor* y en *Mando*. El UUID de avatar en *Visor* y en *Mando* se obtiene al vestirse el respectivo objeto. Es el encargado de dar comienzo a la actividad llamando a los métodos **Start()** de *Cronómetro* y de todos los *Objeto*.

Stop(UUID_avatar_visor) Este método es llamado al pulsar el botón de STOP o al finalizar el tiempo. Si se activa al tocar el botón de STOP se debe tener en cuenta que solo lo puede hacer el avatar que lleve vestido el mando y que tenga el mismo UUID que el atributo */UUID_visor_avatar*. Es el encargado de parar la actividad llamando a los métodos **Reset()** de *Marcador*, de *Cronómetro* y de todos los *Objeto*.También se encarga de registrar en la base de datos */UUID_visor_avatar*, */UUID_mando_avatar*, */puntos* y */tiempo_transcurrido*.

OBJETO

Contiene atributos y métodos para controlar la posición en la que empieza, la posición en la que está, la posición definida como correcta y el movimiento dentro de la casa de cada objeto *Objeto*.

Su atributo */posicion_inicial* sirve para recolocar el objeto al terminar la actividad y su atributo */posicion_final* para comparar la posición actual con la final o correcta.

Sus métodos son los siguientes:

Seleccionar() Este método es llamado cuando el avatar que controla el mando toca el objeto. Se encarga de hacer brillar al objeto para saber que se ha seleccionado correctamente, y se activa el método **Mover(direccion,orientacion)** que permanecía inactivo para *Mando*.

Mover(direccion,orientacion) Este método permanece inactivo hasta que se llame al método **Seleccion()** del *Objeto*. Una vez activo puede ser usado desde *Mando* para mover el *Objeto*. Llama al método **Comprobar_posicion(direccion,orientacion)**, y si cumple las condiciones el objeto se mueve, si no las cumple no hace nada.

Mostrar_posicion() Este método muestra durante un periodo de tiempo un cartel encima del *Objeto*, que pone "korrekt" si la posición actual coincide con */posicion_final* con un margen de error, o "nicht korrekt" si no lo hace.

Comprobar_posicion(direccion,orientacion) Comprueba la *direccion* a la que se le ha ordenado mover el *Objeto*, teniendo en cuenta *orientacion* del avatar y los límites de la casa.

Reset Este método llama a **Mostrar_posicion()** del *Objeto* y **Sumar()** de *Marcador*. Pasado un tiempo mueve el objeto a */posicion_inicial*.

MARCADOR

Contiene métodos y atributos para gestionar el recuento de puntos durante la partida.

Mostrar_puntos() Muestra los puntos guardados en el atributo *puntos*.

Sumar() Suma un punto al atributo *puntos*.

Reset() Llama al método **Mostrar_puntos()** de *Marcador*. Pasado un tiempo pone el atributo *puntos* a cero.

CRONÓMETRO

Contiene métodos y atributos para gestionar el control del tiempo de la actividad.

El atributo *tiempo* tiene por defecto 480 segundos. El atributo *tiempo_transcurrido* se actualiza al terminar la actividad.

Start() Se inicia la cuenta atrás con una variable que contiene el valor del atributo *tiempo*.

Mostrar_tiempo() Muestra el cambio de tiempo.

Reset() Para la cuenta atrás, asignando el tiempo que ha pasado al atributo *tiempo_transcurrido*. Pasado un tiempo se actualiza la variable con el atributo *tiempo*.

VISOR

Contiene atributos y métodos para visualizar cuatro imágenes de la habitación ordenada(controlar la que se desea ver y agrandarla).

El atributo imagen contiene los identificadores de las distintas imágenes que se van a mostrar.

Avatar_UUID() Se activa al vestirse el avatar el *Visor* y tiene como finalidad comunicar el UUID del avatar que lleva vestido el *Visor*.

Cambiar_imagen(lado) Se activa al pulsar el botón izquierdo o derecho del visor de imágenes. Cambia la imagen hacia la izquierda o la derecha dependiendo de *lado*. También se activa al tocar el centro de la imagen y la amplía si está en pequeña o la reduce si está en grande.

MANDO

Contiene atributos y métodos para comunicar la posición a la que debe moverse el objeto seleccionado de la clase *Objeto*. También controla la orientación del avatar necesaria para comunicar la órden de movimiento.

Avatar_UUID() Se activa al vestirse el avatar el *Mando* y tiene como finalidad comunicar el UUID del avatar que lleva vestido el *Mando*.

Pulsar_boton(direccion) Se activa al pulsar uno de los botones del mando con la *direccion* que indica el botón. Llama al método **Mover(direccion,orientacion)** de *Objeto* con la *direccion* obtenida anteriormente y la *orientacion* del avatar en ese momento. Solo se puede realizar en caso de que previamente el avatar haya tocado el *Objeto* y, por tanto, llamado al método **Seleccionar()**.

6.2. Diseño multimedia

6.2.1. Diseño gráfico

En esta subsección debemos diferencias tres tipos de contenido gráfico 3D, uno de ellos de creación propia, otro obtenido de páginas web (Visitar [20] y [17]) desde las que se pueden descargar objetos virtuales e importarlos a *OpenSim* y por último obtenido de páginas web y modificado.

También existe contenido gráfico 2D que se ha introducido en el mundo virtual mediante el uso de texturas. Los gráficos 2D de creación propia se resume a las imágenes y textos que componen los cuadros de los paneles de la actividad 1 y muchas de las texturas usadas en los objetos de la actividad 2 o actividad de ordenar la habitación.
A continuación realizaremos un listado con estos contenidos y se mostraran imágenes de los objetos de creación propia:

Contenido 3D de la sala de entrenamiento o sala 1

Creación propia (6.3) Paneles, imágenes de los paneles, alfombras, cúpula de teletransporte, cartel, botones de Start, pantalla para saber la puntuación. La cúpula que rodea la sala entera es de Francisco Rodríguez. Marcadores y cronómetros tienen texturas proporcionadas por Francisco Rodríguez técnico informático de la UAM .

Extraídos de webs y modificados Sillones.

Extraídos de webs Plantas.

Figura 6.3: Ejemplo de objetos de creación propia de la actividad de entrenamiento o actividad 1.

Contenido 3D de la sala de ordenar la habitación o sala 2

Primero empezaremos por mostrar el diseño del mando (A.9) y el visor de imágenes (A.8), ambos de creación propia.

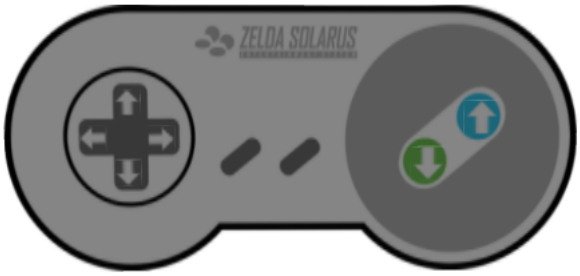


Figura 6.4: Imagen del mando

Figura 6.5: Imágenes del visor

En esta sala las texturas usadas para los posters que se encuentran en las paredes son obtenidas de páginas web.

Creación propia (6.6, 6.7) Posters, cigarrillos, secador, ventilador, lámpara, televisión, estantería, frigorífico, videocámara, cámara de fotos, reloj, auriculares, plato de comida, sandía, gorra, radio, taza de café, aspiradora, lavadora, mechero, copa de vino, ensalada, queso, botella de whisky, fuente, teletransporte, botón de Start, la pantalla para saber la puntuación. Marcadores y cronómetros tienen texturas proporcionadas por Francisco Rodríguez técnico informático de la UAM.

Extraídos de webs y modificados Casa, cama, el contenido del frigorífico.

Extraídos de webs Osito de peluche, mochila, zapatos, silla de escritorio, pastel, helado, rosquillas, barra de pan, mesita, sofá, sillón.

Figura 6.6: Ejemplo de objetos de creación propia de la actividad de ordenar la habitación o actividad 2.

Figura 6.7: Ejemplo de objetos de creación propia de la actividad de ordenar la habitación o actividad 2.

6.2.2. Sonido

El sonido solo se ha introducido en la actividad de entrenamiento o actividad 1. Dentro de esta actividad existen dos tipos de sonidos que está presente en todos los paneles de la sala.

El primer tipo de sonido que podemos diferenciar son audios de la pronunciación en alemán de los textos que aparecen en el lado izquierdo de cada panel. Al seleccionar cualquier cuadro del lado izquierdo de un panel, éste debe mostrar el texto que oculta y reproducir el audio. Todos los audios de este tipo han

sido grabados por la codirectora del proyecto Anke Berns.

La pronunciación es una parte muy importante para todo estudiante de idiomas. Introducir esta pronunciación del concepto provoca en el estudiante, por un lado, que aprenda y mejore sus habilidades auditivas en el idioma objetivo, y por otra parte, fomenta que el estudiante preste mas atención y se sumerja en la actividad.

El segundo tipo de sonido que tenemos son efectos sonoros. Estos audios están relacionados con el final de la actividad y se activan al terminar de jugar en cada panel. Cuando el panel se ha completado entero con éxito, o lo que es lo mismo, cuando el alumno ha obtenido la puntuación máxima del panel que es doce puntos, se emite un sonido de victoria. Por el contrario cuando el alumno termina el panel pero no ha conseguido llegar a doce puntos se emite un sonido de derrota.

Capítulo 7

Implementación del Sistema

En este capitulo se procede a describir y explicar los detalles de implementación del sistema software. También se comentarán los códigos que han tenido mayor complejidad del proceso de desarrollo.

Para consultar el código fuente es necesario acceder al mundo virtual y visualizar los scripts contenidos en los objetos de las actividades. El presente proyecto está alojado en la red de ordenadores de VirtUAM con acceso restringido por las razones que se explicaron en la Introducción, para evitar que personas externas puedan interferir en las actividades. Sin embargo se puede descargar una copia de *OpenSim* con las actividades desde el repositorio de *Google Code*. Esta copia está albergada en un repositorio de tipo *SVN*. Para descargar y acceder al mundo virtual con las actividades véase el apéndice **Manual de Instalación *OpenSim* (C)**.

7.1. Modelo de datos

El lenguaje usado principalmente para implementar las actividades es LSL, aunque también se ha usado el lenguaje OSSL, en menor medida, para complementar algunas funcionalidades que no permite realizar LSL.

Los scripts se programan en el lenguaje LINDEN SCRIPTING LANGUAGE, que se compila a código *.NET*. Dentro de los scripts podemos usar tanto funciones LSL como funciones propias de *OpenSim*, OPEN SIM SCRIPTING LANGUAGE (OSSL).

LINDEN SCRIPTING LANGUAGE (LSL) es un lenguaje de programación orientado a eventos, similar a C y JAVA, creado por Linden inicialmente para dotar de funcionalidad a los objetos y elementos del mundo virtual *Second-Life* y posteriormente empleado por otros como *OpenSim*.

Los scripts están compuestos de [1..N] estados, que pueden contener [1..N] eventos, que a su vez contienen [1..N] funciones. Existe un estado que debe tener obligatoriamente cada script, el `state default{}` o `default{}`

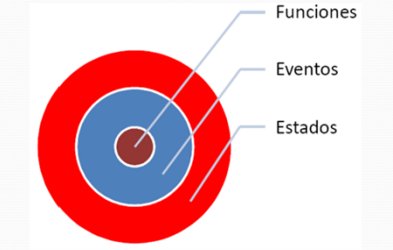


Figura 7.1: Gráfico de contención de la estructura de *LSL*

Los eventos realizados son generados hacia un objeto, como tal, a diferencia de lo que podría ser en un lenguaje como C cuya programación es procedimental y funcional, que de igual manera es un lenguaje muy potente.

Con esto podemos hacer prácticamente lo que deseemos, los scripts en *OpenSim* se emplean para generar partículas y cierto tipos de acciones en objetos.

El siguiente código es un ejemplo de LSL en el que se puede apreciar tanto sintaxis como algunos ejemplos estados, eventos y funciones del lenguaje:

```
// Posicion inicial del objeto;
vector POSICION_INICIAL = <140,132.5,21>;
integer TIEMPO_RESETEO = 5;
float UNIDAD_MOVIMIENTO = 0.5;

default
{
    state_entry()
    {
        llSetPrimitiveParams([PRIM_FULLBRIGHT, ALL_SIDES, FALSE]);
        llSetText("", <1.0, 1.0, 1.0>, 1.0);
        llSetPos(POSICION_INICIAL);
        llListen(CANAL_TIEMPO, "", "", "");
    }
    listen(integer channel, string name, key id, string message)
    {
        if (channel == CANAL_TIEMPO)
        {
            if (message == "start")
            {
                state actividad_empezada;
            }
        }
    }
}
```

En el ejemplo se puede apreciar como se declaran tres tipos diferentes de variables *vector, integer y float*. Luego de esto, tenemos el estado *default* que está compuesto por dos eventos. El primer evento que aparece es *state_entry*, que es el evento que cualquier estado ejecuta en primer lugar. Dentro de *state_entry* existen cuatro funciones de LSL. El siguiente evento es *listen*, que sirve para "oir" por los canales que tengamos abiertos. Este evento junto con las funciones, llSay o llShout entre otras, nos proporcionan la

forma en la que se realiza la comunicación entre los diferentes scripts.

7.2. Control de versiones

Para controlar los cambios realizados en las actividades a lo largo del proyecto se ha usado *TortoiseSVN* en un entorno de desarrollo *Windows*. Este sistema de versiones posee las siguientes características

Integración con el shell de *Windows* *TortoiseSVN* se integra perfectamente en el shell de *Windows*. Esto significa que puede seguir trabajando con las herramientas que ya conoce. Y que no tiene que cambiar a una aplicación diferente cada vez que necesite las funciones del control de versiones.

Y ni siquiera está obligado a usar el Explorador de *Windows*. Los menús contextuales de *TortoiseSVN* también funcionan en otros administradores de archivos, y en el diálogo Fichero/Abrir que es común a la mayoría de aplicaciones estándar de *Windows*. Sin embargo, debe tener en cuenta que *TortoiseSVN* está desarrollado con la mirada puesta en hacerle extensión del Explorador de *Windows*. Por este motivo, puede que en otras aplicaciones la integración no sea tan completa.

Iconos sobreimpresionados El estado de cada carpeta y fichero versionado se indica por pequeños iconos sobreimpresionados. De esta forma, puede ver fácilmente el estado en el que se encuentra su copia de trabajo.

Fácil acceso a los comandos de *Subversion* Todos los comandos de *Subversion* están disponibles desde el menú contextual del explorador. *TortoiseSVN* añade su propio submenú allí.

Versionado de carpetas CVS sólo controla la historia de ficheros individuales, pero *Subversion* implementa un sistema "virtual" de ficheros versionados que sigue la pista de los cambios en todos los árboles de directorios en el tiempo. Los ficheros y los directorios están versionados. Como resultado, hay comandos reales en el lado del cliente como mover y copiar que operan en ficheros y directorios.

Confirmaciones atómicas Una confirmación o bien entra en el repositorio completamente, o no entra en absoluto. Esto permite a los desarrolladores construir y confirmar cambios como unidades lógicas.

Metadatos versionados Cada fichero y directorio tiene un conjunto invisible de "propiedades" adjuntos. Puede inventarse y almacenar cualquier par de clave/valor que desee. Las propiedades se versionan en el tiempo, igual que el contenido de los ficheros.

Elección de capas de red *Subversion* tiene una noción abstracta del acceso al repositorio, haciendo que la gente pueda implementar nuevos mecanismos de red fácilmente. El "avanzado" servidor de red de *Subversion* es un módulo para el servidor web Apache, que habla una variante de HTTP llamada WebDAV/DeltaV. Esto dota a *Subversion* una gran ventaja en estabilidad e interoperatividad, y proporciona varias características importantes gratis: autentificación, autorización, compresión de la transmisión y navegación del repositorio, por ejemplo. También está disponible un proceso servidor de *Subversion* independiente. Este servidor habla un protocolo propio que puede encapsularse fácilmente sobre ssh.

Manejo de datos consistente *Subversion* expresa las diferencias entre ficheros usando un algoritmo de diferenciación binario, que funciona exactamente igual tanto en ficheros de texto (legibles por los humanos) como en ficheros binarios (que no son legibles por nosotros). Ambos tipos de ficheros

se almacenan igualmente comprimidos en el repositorio, y las diferencias se transmiten en ambas direcciones por la red.

Etiquetado y creación de ramas eficiente El coste de crear una rama o una etiqueta no necesita ser proporcional al tamaño del proyecto. *Subversion* crea ramas y etiquetas simplemente copiando el proyecto, utilizando un mecanismo similar a los vínculos duros. Por tanto estas operaciones llevan un tiempo pequeño y constante, y muy poco espacio en el repositorio.

Extensibilidad *Subversion* no tiene lastre histórico; está implementado como una colección de librerías C compartidas con APIS bien definidas. Esto hace que *Subversion* sea extremadamente mantenible y se pueda utilizar por otras aplicaciones y lenguajes.

7.3. Implementación Panel

Este es el código principal de la actividad 1 o actividad de entrenamiento. Desde el que se manejan todas las funciones de los paneles. Todos las variables *CANAL_x* son los canales por los que enviamos un mensaje codificado para comunicarnos con los demás scripts. Por cada canal se envía información diferente.

La función `ListCompare(list a, list b)` lo que hace es comparar dos listas de caracteres. Esta función se utiliza para comparar el atributo *identificador* que ha llegado por CANAL_PANEL con una lista vacía. En caso de que no haya ningún **Cuadro** mostrado, se asigna el identificador a una variable para más tarde poder comparar los identificadores cuando se muestre otro **Cuadro** del lado opuesto al primero. Cuando se comparen los dos identificarse, es decir que haya dos **Cuadro** mostrados, se concede un punto si acierta o se ocultan los **Cuadro** si se falla. Ambas acciones se comunican a marcador por CANAL_PANEL.

Si llega el mensaje “stop” por el CANAL_TIEMPO significa que la actividad se ha acabado, ya sea por límite de tiempo o porque el jugador ha parado la actividad. En ambos casos se manda a la base de datos externa a la de *OpenSim* el nombre del jugador y su puntuación.

```
1  // Canal de comunicaci\'{o}n del primer panel.
2  integer CANAL_PANEL = 411;
3  // Canal reset.
4  integer CANAL_TIEMPO = 440;
5
6  integer CANAL_UUID = 490;
7
8  string UUID;
9
10 integer puntos;
11
12 // Nombre de la pareja a traves del que comprobamos si acierta o falla.
13 list nombre_pareja = [];
14
15 integer ListCompare(list a, list b) {
16     integer aL = a != [];
17     if (aL != (b != [])) return 0;
18     if (aL == 0)         return 1;
19
20     return !llListFindList((a = []) + a, (b = []) + b);
```

```
21  }
22
23  default
24  {
25      state_entry()
26      {
27          llListen(CANAL_PANEL, "", "", "");
28          llListen(CANAL_TIEMPO, "", "", "");
29          llListen(CANAL_UUID, "", "", "");
30          nombre_pareja = [];
31      }
32      listen(integer channel, string name, key id, string message)
33      {
34          if (channel == CANAL_PANEL)
35          {
36              // Si no se ha destapado ninguna casilla.
37
38              list nombre=llCSV2List(message);
39              //Comprobamos si existe alg\'{u}n {\bf Cuadro} descubierto, y
                    si lo hay comprobamos
40              //si es correcto o no
41              if( ListCompare(nombre_pareja,[]) )
42              {
43                  nombre_pareja = nombre;
44              }
45              else
46              {
47                  llSleep(1);
48                  // Si acierta
49                  if (llList2String(nombre,0) == llList2String(nombre_pareja
                        ,0) && llList2String(nombre,1) !=llList2String(
                        nombre_pareja,1))
50                  {
51                      llShout(CANAL_PANEL, "acierto");
52                      puntos++;
53                  }
54                  // Si falla.
55                  else
56                  {
57                      llShout(CANAL_PANEL, "fallo");
58                  }
59                  nombre=[];
60                  nombre_pareja = [];
61              }
62          }
63          else if (channel == CANAL_UUID)
64          {
65              UUID = message;
66          }
67          else if (channel == CANAL_TIEMPO)
68          {
69              if(message == "stop")
70              {
71              //Mandamos nombre y puntuaci\'{o}n a la base de datos
72                  llHTTPRequest( "http://127.0.0.1:8080/PFCPrueba/escritura.
                        jsp?"+
```

```
                                "nombre="+ (string)llKey2Name(UUID) +
                                "&panel="+ "4" +
                                "&puntos="+ (string)puntos
                                , [HTTP_METHOD,"GET"], "");
                    UUID = "";
                    nombre_pareja = [];
                    puntos = 0;
                }
            }
        }
    }
```

7.4. Implementación movimiento objetos

El fragmento de código que se muestra al final de esta sección ha sido el de mayor complejidad. Para poder comprender porque es tan complejo, primero se debe explicar varias cosas.

Un objeto tiene 4 zonas predefinidas desde donde podría ser controlado (véase 7.2), en relación a la posición que tenga el avatar con respecto a este objeto: frente al objeto, detrás del objeto, a la derecha del objeto y a la izquierda del objeto.

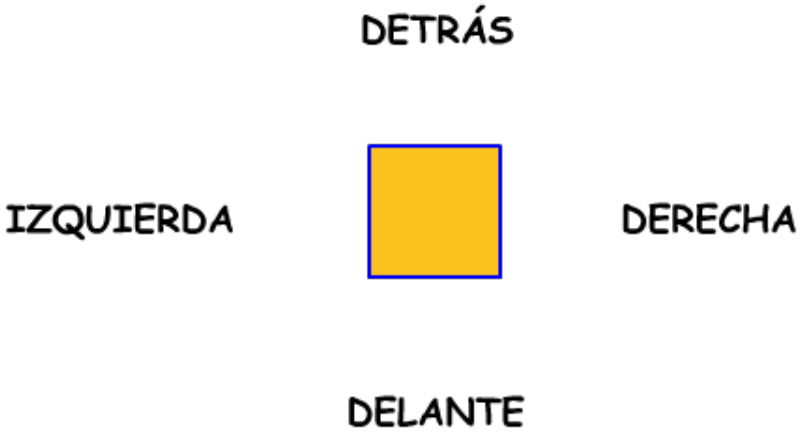


Figura 7.2: Posiciones relativas de un avatar respecto a un objeto fijo.

Dependiendo de la zona, el avatar tiene que colocar la cámara (campo visual, véase 7.3) de forma diferente.

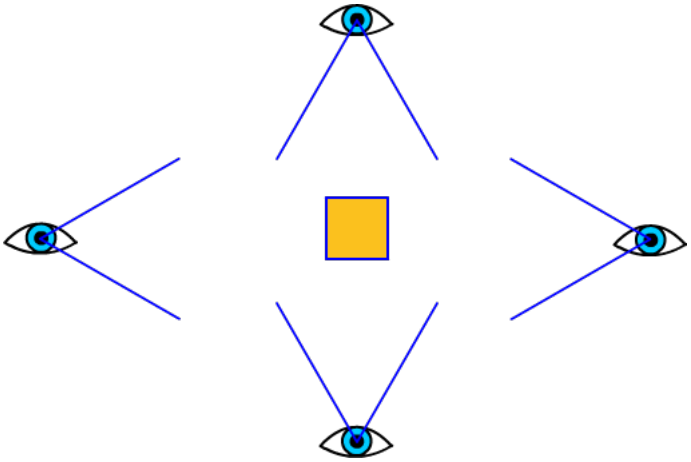


Figura 7.3: Rotación de la cámara del avatar tomando un punto de referencia fijo.

La primera versión del script que se hizo solo funcionaba correctamente desde la posición "DELANTE.[en] la 7.2. El script no detectaba la posición en la que se encontraba el avatar por lo cual el objeto se controlaba como si el avatar estuviese siempre delante del objeto. Un ejemplo de cómo actuaba: si el avatar se situaba detrás del objeto pulsando el botón de la derecha, el objeto se movía hacia la izquierda en lugar de moverse hacia la derecha.

En la segunda versión del script se optó por la posición relativa del avatar respecto al objeto. De esta forma se conseguía controlar el objeto, en las diferentes “zonas” (véase, 7.3), de manera correcta. Pero de esta forma surgía un nuevo fallo, porque el script estaba hecho de forma que si la posición del eje “y” del avatar era mayor o menor que la del objeto se situaba delante o detrás en una primera aproximación (véase, 7.4, rojo y azul), y una vez situado se comparaba con la del eje “x” entre el avatar y el objeto (véase 7.4, verde). Si resultaba, la diferencia en valor absoluto de las posiciones, mayor en el eje x, el avatar estaba en el lado derecho o izquierdo, y si era mayor en el eje “y” estaba delante o detrás dependiendo del resultado del eje “y” (véase 7.4, verde).

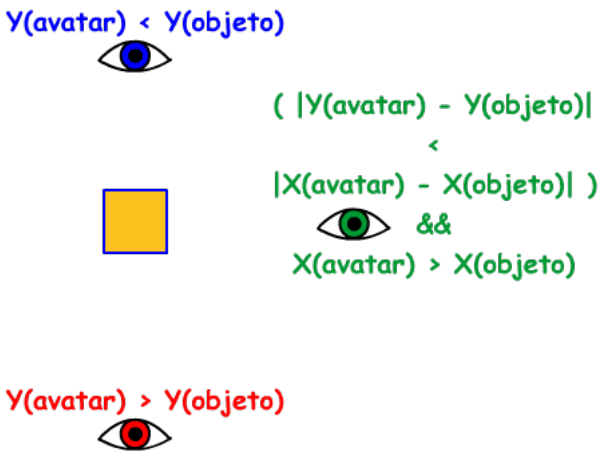


Figura 7.4: Fórmula usadas para obtener la posición relativa del avatar respecto al objeto.

La tercera versión del script se hizo por la rotación que tenía la cámara del avatar y no por su posición.

Si tenemos en cuenta que la rotación de la cámara es la que va a influir en como vamos a manejar el objeto que estamos visualizando, deja de tener sentido el tomar como referencia la posición del avatar o la del objeto. Por ello vemos primero la rotación de la cámara del avatar y si está dentro de uno de los 4 rangos marcados, entonces cambiamos a una u otra zona de movimiento del mando (Véase 7.5). Los rangos en grados son: DELANTE (135º,225º), DETRáS (315º,45º), IZQUIERDA (45º,135º) y DERECHA (225º,315º).

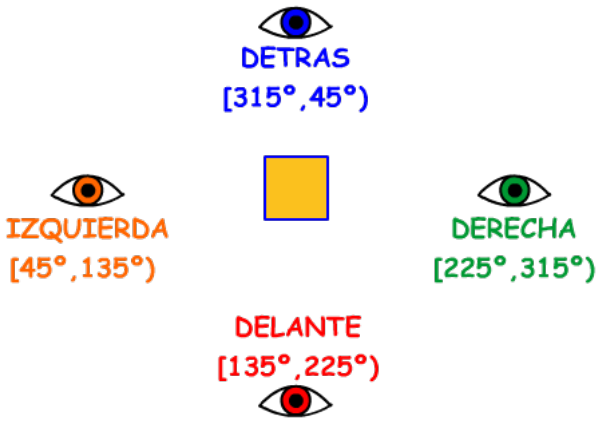


Figura 7.5: Rotación de la cámara del avatar al visualizar el objeto.

A parte de que hubo varias versiones había otro gran problema con la tercera versión y es que en LSL no existe concentración de código. Por tanto, cada vez que se reescribe el código en cualquiera de los otros scripts que interactúan con un objeto, este script se debe adaptar a esa modificación, y modificar uno a uno los 31 *Objetos*.

El script se implemento para que dependiendo de la "dirección" que se pulsaba en el mando y la "orientacion" de la cámara del avatar que controlaba el mando, el *Objeto* se moviera correctamente.

```
1  state objeto_activo
2  {
3      state_entry()
4      {
5          llShout(CANAL_UUIDS, llGetKey());
6          llSetPrimitiveParams([PRIM_FULLBRIGHT, ALL_SIDES, TRUE]);
7          llListen(CANAL_UUIDS, "", "", "");
8          llListen(CANAL_MOVIMIENTO, "", "", "");
9          llListen(CANAL_TIEMPO, "", "", "");
10         }
11         listen(integer channel, string name, key id, string message)
12         {
13             if(channel==CANAL_MOVIMIENTO)
14             {
15                 list movimiento=llCSV2List(message);
16                 vector posobjeto=llGetPos();
17
18                 if (llList2String(movimiento,0) == "arriba" && valida(
                       llGetPos()+<0,0,UNIDAD_MOVIMIENTO>))
19                 {
```

```
20                      llSetPos(llGetPos()+<0,0,UNIDAD_MOVIMIENTO>);
21                      //llMoveToTarget(llGetPos()+<0,0,UNIDAD_MOVIMIENTO>,
                            0.05);
22                  }
23                  else if (llList2String(movimiento,0) == "abajo" && valida(
                        llGetPos()-<0,0,UNIDAD_MOVIMIENTO>))
24                  {
25                      llSetPos(llGetPos()-<0,0,UNIDAD_MOVIMIENTO>);
26                      //llMoveToTarget(llGetPos()-<0,0,UNIDAD_MOVIMIENTO>,
                            0.03);
27                  }
28                  else if(((float)llList2String(movimiento,1))>=((float)135)
                        && ((float)llList2String(movimiento,1))<((float)225))//
                        Estoy delante del objeto
29                  {
30                      if (llList2String(movimiento,0) == "izquierda" &&
                            valida(llGetPos()+<UNIDAD_MOVIMIENTO,0,0>) )//
                            Izquierda
31                      {
32                          llSetPos(llGetPos()+<UNIDAD_MOVIMIENTO,0,0>);
33                      }
34                      else if (llList2String(movimiento,0) == "derecha" &&
                            valida(llGetPos()-<UNIDAD_MOVIMIENTO,0,0>))//Derecha
35                      {
36                          llSetPos(llGetPos()-<UNIDAD_MOVIMIENTO,0,0>);
37                      }
38                      else if (llList2String(movimiento,0) == "delante" &&
                            valida(llGetPos()-<0,UNIDAD_MOVIMIENTO,0>))//Delante
39                      {
40                          llSetPos(llGetPos()-<0,UNIDAD_MOVIMIENTO,0>);
41                      }
42                      else if (llList2String(movimiento,0) == "detras" &&
                            valida(llGetPos()+<0,UNIDAD_MOVIMIENTO,0>))//Detras
43                      {
44                          llSetPos(llGetPos()+<0,UNIDAD_MOVIMIENTO,0>);
45                      }
46                  }
47                  else if(((float)llList2String(movimiento,1))>=((float)45)
                        && ((float)llList2String(movimiento,1))<((float)135))
48                  {//Estoy a la izquierda del objeto
49                      if (llList2String(movimiento,0) == "detras" && valida(
                            llGetPos()+<UNIDAD_MOVIMIENTO,0,0>))//Izquierda
50                      {
51                                      llSetPos(llGetPos()+<UNIDAD_MOVIMIENTO
                                            ,0,0>);
52                      }
53                      else if (llList2String(movimiento,0) == "delante" &&
                            valida(llGetPos()-<UNIDAD_MOVIMIENTO,0,0>))//Derecha
54                      {
55                                      llSetPos(llGetPos()-<UNIDAD_MOVIMIENTO
                                            ,0,0>);
56                      }
57                      else if (llList2String(movimiento,0) == "izquierda" &&
                            valida(llGetPos()-<0,UNIDAD_MOVIMIENTO,0>))//Delante
58                      {
59                                      llSetPos(llGetPos()-<0,
```

```
                                        UNIDAD_MOVIMIENTO,0>);
60              }
61              else if (llList2String(movimiento,0) == "derecha" &&
                    valida(llGetPos()+<0,UNIDAD_MOVIMIENTO,0>))//Detras
62              {
63                              llSetPos(llGetPos()+<0,
                                    UNIDAD_MOVIMIENTO,0>);
64              }
65          }
66          else if(((float)llList2String(movimiento,1))>=((float)225)
                && ((float)llList2String(movimiento,1))<((float)315))//
                Estoy a la derecha del objeto
67          {
68              if (llList2String(movimiento,0) == "delante" && valida(
                    llGetPos()+<UNIDAD_MOVIMIENTO,0,0>))//Izquierda
69              {
70                              llSetPos(llGetPos()+<UNIDAD_MOVIMIENTO
                                    ,0,0>);
71              }
72              else if (llList2String(movimiento,0) == "detras" &&
                    valida(llGetPos()-<UNIDAD_MOVIMIENTO,0,0>))//Derecha
73              {
74                              llSetPos(llGetPos()-<UNIDAD_MOVIMIENTO
                                    ,0,0>);
75              }
76              else if (llList2String(movimiento,0) == "derecha" &&
                    valida(llGetPos()-<0,UNIDAD_MOVIMIENTO,0>))//Delante
77              {
78                              llSetPos(llGetPos()-<0,
                                    UNIDAD_MOVIMIENTO,0>);
79              }
80              else if (llList2String(movimiento,0) == "izquierda" &&
                    valida(llGetPos()+<0,UNIDAD_MOVIMIENTO,0>))//Detras
81              {
82                              llSetPos(llGetPos()+<0,
                                    UNIDAD_MOVIMIENTO,0>);
83              }
84          }else if(((float)llList2String(movimiento,1))>=((float)315)
                || ((float)llList2String(movimiento,1))<((float)45))//
                Estoy detras del objeto
85          {
86              if (llList2String(movimiento,0) == "derecha" && valida(
                    llGetPos()+<UNIDAD_MOVIMIENTO,0,0>))//Izquierda
87              {
88                          llSetPos(llGetPos()+<UNIDAD_MOVIMIENTO
                                ,0,0>);
89              }
90              else if (llList2String(movimiento,0) == "izquierda" &&
                    valida(llGetPos()-<UNIDAD_MOVIMIENTO,0,0>))//Derecha
91              {
92                          llSetPos(llGetPos()-<UNIDAD_MOVIMIENTO
                                ,0,0>);
93              }
94              else if (llList2String(movimiento,0) == "detras" &&
                    valida(llGetPos()-<0,UNIDAD_MOVIMIENTO,0>))//Delante
95              {
```

```
96                                  llSetPos(llGetPos()-<0,UNIDAD_MOVIMIENTO
                                        ,0>);
97                          }
98                          else if (llList2String(movimiento,0) == "delante" &&
                                valida(llGetPos()+<0,UNIDAD_MOVIMIENTO,0>))//Detras
99                          {
100                                 llSetPos(llGetPos()+<0,UNIDAD_MOVIMIENTO
                                        ,0>);
101                         }
102                 }
103             }
104             else if (channel == CANAL_UUIDS)
105             {
106                 if(llGetKey()!=message)
107                 {
108                         state actividad_empezada;
109                 }
110             }
111             else if (channel == CANAL_TIEMPO)
112             {
113                 if (message == "stop")
114                 {
115                         llSetTimerEvent(TIEMPO_RESETEO);
116                 }
117             }
118         }
119         timer()
120         {
121             llSetTimerEvent(0);
122             state default;
123         }
124     }
```

7.5. Herramientas usadas

7.5.1. LaTeX

LaTeXes un sistema de composición de textos, orientado a la creación de textos científicos y técnicos en especial.

LaTeXestá formado por un gran conjunto de macros de TeX, escrito por Leslie Lamport en 1984, con la intención de facilitar el uso del lenguaje de composición tipográfica, TeX. LaTeXse extendió rápidamente por todo el sector científico y técnico gracias a su facilidad de uso y toda la potencia de TeX.

Su código abierto permitió que muchos usuarios realizasen nuevas utilidades que extendiesen sus capacidades con objetivos muy variados, apareciendo "dialectos" de LaTeX, muchas veces incompatibles entre sí. En 1993 se anunció una reestandarización completa de LaTeXpara evitar discrepancias anteriores, creándose nuevas extensiones como la posibilidad de escribir transparencias por ejemplo.

La característica más relevante que se creó fue la arquitectura modular. Se estableció un núcleo central, el compilador, que mantiene las funcionalidades de la versión anterior pero permite incrementar su potencia y versatilidad por medio de diferentes paquetes, que cualquiera puede crear uno nuevo, que sólo

se cargan si son necesarios.

Esta memoria está desarrollada con LaTeX. Además del diseño limpio y claro que proporciona al documento de manera automática, crea índices, referencias, bibliografía, ajusta figuras y listados, etc., y un sinfín de características que facilitan la labor generando además un documento impecable.

7.5.2. Dia

Dia es una aplicación informática de propósito general para la creación de diagramas, desarrollada como parte del proyecto GNOME. Está concebido de forma modular, con diferentes paquetes de formas para diferentes necesidades.

Dia está diseñado como un sustituto de la aplicación comercial Visio de Microsoft. Se puede utilizar para dibujar diferentes tipos de diagramas. Actualmente se incluyen diagramas entidad-relación, diagramas UML, diagramas de flujo, diagramas de redes, diagramas de circuitos eléctricos, etc. Nuevas formas pueden ser fácilmente agregadas, dibujándolas con un subconjunto de SVG e incluyéndolas en un archivo XML.

El formato para leer y almacenar gráficos es XML (comprimido con gzip, para ahorrar espacio). Puede producir salida en los formatos EPS, SVG y PNG.

7.5.3. GIMP

GIMP (GNU Image Manipulation Program) es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. Es un programa libre y gratuito. Está englobado en el proyecto GNU y disponible bajo la Licencia pública general de GNU.

GIMP lee y escribe la mayoría de los formatos de ficheros gráficos, entre ellos jpg, gif, png, pcx, tiff, y los de Photoshop, además de poseer su propio formato de almacenamiento de ficheros, xcf. También es capaz de importar ficheros en pdf y también imágenes vectoriales en formato svg creadas, por ejemplo, con Inkscape.

El uso que se le ha dado a esta herramienta en el proyecto engloba muchos campos. A continuación detallamos una lista con los usos que se le han dado:

- Crear texturas que posteriormente se importarían a *OpenSim*.
- Modificar todas las imágenes de los paneles de la actividad 1, para hacerlas del mismo tamaño y visualizar mejor su contenido.
- Crear todos los textos de los textos de los paneles de actividad 1.
- Modificar todas las imágenes que se han incluido en esta memoria y en la wiki de Google Code ([15])

7.6. Audacity

Audacity es una aplicación informática multiplataforma libre, que se puede usar para grabación y edición de audio, fácil de usar, distribuido bajo la licencia GPL.

Fue creado en otoño de 1999 por Dominic Mazzoni y Roger Dannenberg en la universidad de Carnegie Mellon. Tras lo cual fue publicado en SourceForge.net como software libre en mayo de 2000. En mayo de 2008, *Audacity* fue incorporado a la lista de los 100 mejores productos del año según la los lectores y editores de la revista PC World.

Entre sus características principales se encuentran:

- Grabación de audio en tiempo real.
- Edición archivos de audio tipo Ogg Vorbis, MP3, WAV, AIFF, AU , LOF y WMP.
- Conversión entre formatos de tipo audio.
- Importación de archivos de formato MIDI,RAW y MP3.
- Edición de pistas múltiples.
- Agregar efectos al sonido (eco, inversión, tono, etc).
- Posibilidad de usar plug-ins para aumentar su funcionalidad.

En el proyecto se ha mencionado que se han usado audios, como por ejemplo en los paneles de la actividad 1. *Audacity* se ha usado principalmente para mejorar la calidad de audio de la pronunciación de objetos y conceptos. También se ha editado la duración de estos audios debido a que tenían muchos intervalos de grabación vacíos de contenido.

7.7. Imprudence Viewer

Imprudence es un visor de código abierto basado en el código del visor de *Second Life* (diseñado principalmente para su uso en *OpenSim*, pero compatible con *Second Life*). Es el visor más innovador y su principal interés es mejorar la usabilidad a través de la experiencia del usuario y un entorno moderno. Permite importar fácilmente al mundo virtual texturas, sonidos y modelos 3D.

Tras la experiencia con los distintos visores a lo largo del proyecto, se recomienda el uso de este visor. Esta herramienta se usa para acceder y visualizar el mundo virtual que se crea con *OpenSim*.

7.8. RealXtend Viewer

RealXtend tiene una versión propia de *OpenSim* y visores adaptados para renderizados y modelados en 3D; su trabajo está muy centrado en este aspecto, lo que hace que los requerimientos gráficos en el ordenador del usuario sean superiores.

Al principio se empezó a usar esta herramienta para acceder al mundo virtual, pero comenzó a dar fallos y se buscaron navegadores alternativos, hasta encontrar *Imprudence*.

Capítulo 8

Pruebas del Sistema

En este capítulo se hablará de las pruebas que se han realizado a las actividades desarrolladas. La fase de pruebas es una parte importante en todo desarrollo software. El objetivo de las pruebas es la verificación de que el proyecto cumple con los requisitos especificados. Existen diferentes enfoques a la hora de realizar las pruebas de software.

Este proyecto es una mezcla entre un videojuego online y una aplicación de gestión. Por ello deberemos de hacer varios enfoques a la hora de hacer las pruebas de nuestro sistema software, a saber:

- **Pruebas unitarias:** Permiten probar el correcto funcionamiento de un módulo de código. De esta forma conseguimos saber que cada uno de los módulos que integran el sistema software funcionan correctamente por separado.
- **Pruebas de integración:** Se realizan en el ámbito del desarrollo software una vez aprobadas las pruebas unitarias. Se refiere a la prueba de todos los elementos unitarios que componen un proceso realizado en conjunto. De esta forma conseguimos verificar que las partes de un sistema software funcionan de forma conjunta.
- **Pruebas funcionales:** Basadas en la ejecución, revisión y retroalimentación de las funcionalidades diseñadas para el software.
- **Pruebas de usabilidad:** Pruebas que se encargan de medir en que grado puede una persona usar el sistema. Consiste en seleccionar un grupo de usuarios y solicitarles que lleven a cabo las tareas para las cuales fue diseñada la aplicación, mientras que el equipo de desarrollo toma nota para evaluar la respuesta del usuario.

Para la realización de todas las pruebas realizadas en ambas actividades ha sido necesaria la intervención de un grupo de usuarios que interactúen con la aplicación. En este grupo se encontraban la codirectora del proyecto Anke Berns y el técnico informático Francisco Rodríguez de la Universidad Autónoma de Madrid.

8.1. Pruebas unitarias

A medida que se iba realizando la implementación de los módulos que compondrían las actividades, se iban realizando las pruebas unitarias de cada componente de forma no automatizada.

Como se ha comentado anteriormente, se ha seguido una metodología ágil para el desarrollo de las actividades, por lo que las pruebas unitarias se iban realizando cada vez que se terminaba algún módulo de las actividades, en los primeros dias del tiempo de implementación de cada actividad (Diagrama de Gantt del capítulo 4 4.1).

8.2. Pruebas de integración

Según se iban desarrollando las diferentes funcionalidades del proyecto y sus módulos iban cumpliendo los requisitos software a través de las pruebas unitarias, se iba comprobando que la integración de los mismos en el software haciendo cumplía con los requisitos recolectados en la fase de análisis.

Estas pruebas se realizaban al finalizar la semana sobre los módulos que habían sido integrados durante este periodo de tiempo.

8.3. Pruebas funcionales

Una vez que habíamos comprobado que la integración de los módulos no remitía ningún tipo de error, ni comportaba ningún tipo de comportamiento extraño en los módulos ya testeados, se dispuso a comprobar que el software desarrollado cumplía con los requisitos funcionales.

8.4. Pruebas de usabilidad

Dado el uso específico de las actividades desarrollas se ha pedido a varios sujetos que prueben la actividad.

En la actividad 1 o actividad de entrenamiento, debían conseguir obtener la máxima puntuación en cada panel, y posteriormente debían consultar su puntuación y usar el teletransporte a la segunda sala.

En la actividad 2 o actividad de ordenar la habitación se pedía, que realizasen la actividad dentro del límite de tiempo sin restricción de puntuación. Al finalizar la actividad debían consultar la puntuación en la pantalla habilitada para ello y usar el teletransporte.

Capítulo 9

Conclusiones

El presente proyecto fue diseñado dentro de la plataforma VirtUAM, y esta enfocado al aprendizaje de lenguas extranjeras. Se utilizó la plataforma para que los investigadores, entre los que se encuentra la codirectora del proyecto Anke Berns, puedan explorar el potencial educativo y motivacional de los mundos virtuales en combinación con aplicaciones parecidas a videojuegos. Esta idea de combinar mundos virtuales y videojuegos se explicó en el capítulo de Introducción de esta memoria.

Las aplicaciones desarrolladas presentan aún algunas limitaciones, como la la falta del chat por voz en lugar de chat escrito, o el diseño de actividades basadas en los principios de aprendizaje adaptativos.

Por otra parte mediante la inclusión de feedback (tanto los audios de la pronunciación en alemán como las puntuaciones obtenidas) hacen que los estudiantes se sientan motivados durante todo el proceso de aprendizaje mientras están jugando. Esto se puede explicar por el hecho de que el juego es a menudo percibido por los estudiantes como diversión y entretenimiento, más que algo relacionado con el aprendizaje.

Esta herramienta solo se puede usar como complemento a la enseñanza presencial, y en ningún momento puede sustituirla.

Los juegos hacen el proceso de aprendizaje más fácil y rápido, debido a varios factores, como por ejemplo que proporcionan feedback en tiempo real. El entorno inmersivo del juego y el hecho de que el vocabulario se presenta en el contexto hace que sea más fácil de entender y aprender.

Los estudiantes pueden realizar las mismas, e incluso más actividades a través del mundo virtual, ya que permite un entorno de aprendizaje como en la enseñanza presencial, con la diferencia de que la interacción en el juego se percibe libre de miedo al fracaso.

Personalmente, la elaboración tanto del proyecto como de la documentación ha supuesto un gran reto. Es una nueva experiencia que he tenido que afrontar sin haber hecho nada parecido hasta el momento. Los trabajos realizados en años anteriores, de las diferentes asignaturas de la carrera, han sido de menor duración y dificultad. Enfrentarte a un un proyecto de estas características supone un cambio bastante importante en la perspectiva de la carrera, porque supone vivir a pequeña escala la experiencia del mundo laboral. Hasta ahora, lo visto en las asignaturas estaba principalmente enfocado a adquirir conocimientos teóricos y prácticos, pero siempre los profesores sabían que tenías que hacer y como tenías que hacerlo. Este proyecto ha supuesto tener que construir desde una idea una herramienta que no se sabía en un principio como iba a funcionar, pero que poco a poco fue cogiendo forma hasta obtener el producto final.

También ha sido difícil aprender como funcionan las tecnologías, herramientas y los lenguajes de programación usados en el proyecto, que no conocía hasta el momento. Aprender a manejarlos y sacarles provecho ha supuesto perder el miedo a trabajar con herramientas y tecnologías desconocidas.

Aunque el proyecto me ha acercado al mundo laboral, siempre he tenido ayuda por parte de la codirectora y promotora del proyecto Anke Berns, y el técnico en informática de la Universidad Autónoma de Madrid, Francisco Rodriguez, junto con el codirector del proyecto Daniel Molina y el profesor Manuel Palomo. Les estoy tremendamente agradecidos porque me han prestado toda la ayuda posible, y han hecho que la elaboración del proyecto resulte más fácil.

Apéndices

Apéndice A

Manual de usuario

A.1. Acciones Básicas

Las acciones más básicas que podemos realizar son:

- **Moverse**: Para ello hay que utilizar las flechas del teclado. También se pueden usar las teclas “w”, “a”, “s”, “d”. Para correr pulsa dos veces sobre la dirección a la que deseas dirigirte.
- **Hablar**: En la parte inferior aparece un cuadro de texto destinado al chat, lugar en el que podemos escribir para comunicarnos con otros avatares.
- **Volar**: Al pulsar en el menú inferior esta opción podremos desplazarnos de forma más rápida por el mundo virtual, para volver al suelo pulsa Dejar de volar.
- **Sentarse**: Para ello hay que hacer clic sobre la silla en la que queremos ocupar asiento o clic derecho y Sentarse aquí en el menú. Para levantarse hay que pulsar Levantarse.
- **Tocar**: Para tocar objetos, simplemente hay que hacer clic sobre ellos.

A.2. Inventario

El inventario (Figura A.1) es un elemento importante para el usuario dentro del mundo virtual en el que se alojan todos los ítems que posee el usuario (texturas, sonidos, . . .). Lo podemos localizar en el menú inferior, al hacer clic aparecerá una ventana en la que podemos ver una estructura de carpetas.

A.2.1. Estructura

En el inventario de un avatar podemos encontrar las siguientes carpetas y secciones (Véase figura A.1):

- **Animations**: Serie de animaciones por defecto que trae OpenSim y que pueden ser ejecutadas dentro del mundo virtual.
- **Body parts**: Esta es una sección importante en la que crearemos aquellas partes del cuerpo y ropa que queremos personalizar.
- **Landmarks**: Elementos que guardan una posición determinada, de esta forma tenemos el acceso directo a dicha posición con un simple doble clic.
- **Notecards**: Archivos de texto plano.

- **Objects**: Lugar donde se guardan los objetos que importamos o cogemos del entorno.
- **Scripts**: En esta carpeta podemos crear y guardar aquellos scripts de importancia a los que queremos tener acceso en cualquier momento.
- **Sounds**: Aquí se guardan los sonidos que importamos.
- **Texturas**: Carpeta dedicada a las texturas importadas.
- **Trash**: Papelera en la que se guardan los elementos eliminados.

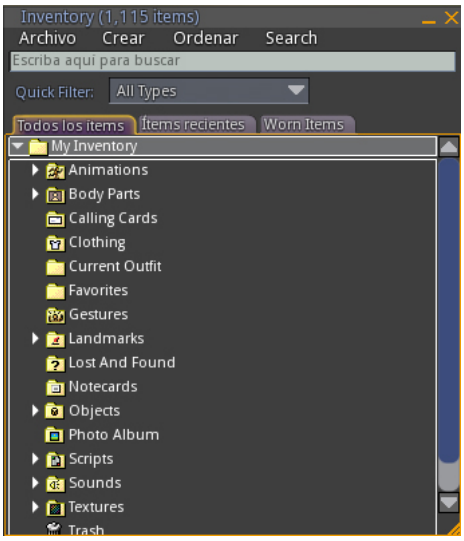


Figura A.1: Inventario de un avatar.

A.2.2. Vistiendo objetos

A veces, es de gran utilidad anexar objetos que tenemos en el inventario al avatar. Para ello al hacer clic derecho sobre el objeto, nos aparece un menú, en el que en la parte inferior encontramos:

- *Ponerse*: El avatar se viste el objeto, en la posición por defecto o en la última posición utilizada.
- *Anexar a*: Para vestir el objeto a una parte del cuerpo (brazo, pierna, cabeza . . .).
- *Anexar como HUD*: Si utilizamos esta opción el objeto aparecerá directamente en la pantalla en la posición seleccionada.

Para separar el objeto del avatar, hacemos doble clic sobre el objeto del inventario, o clic derecho directamente sobre el objeto y *Detach*.

A.3. Actividad 1

Para la correcta realización de esta actividad es necesario que el avatar esté sentado en el sillón correspondiente al panel que va a utilizar (Véase A.2). Para sentarse solo debe tocar el sillón o seleccionarlo con el botón derecho del ratón y elegir la opción sentarse.

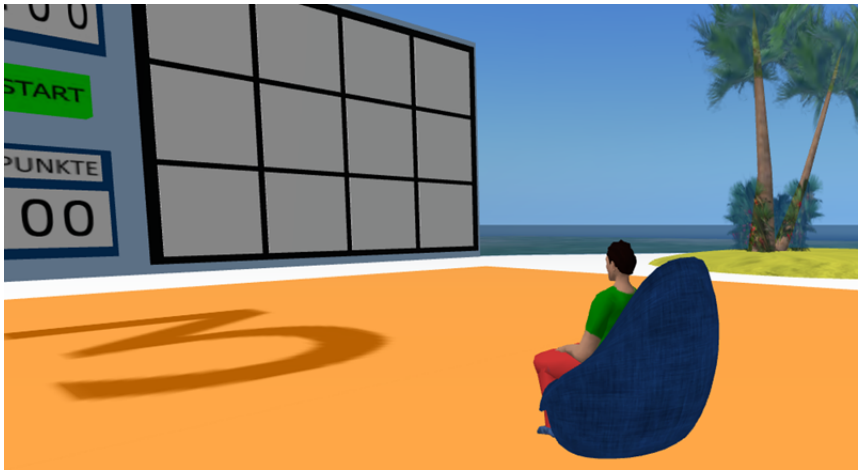


Figura A.2: Imagen de un avatar sentado antes de comenzar la actividad.

Una vez esté el avatar sentado debe tocar el botón START para dar comienzo al juego. Al tocar START, verá que el tiempo comienza a contar y que el botón que antes era verde y ponía START en el centro, ahora ha cambiado y es rojo y pone STOP.

A.3.1. Juego

El panel está formado por veinticuatro cuadros, la mitad a la izquierda del botón de START/STOP, y la otra mitad a la derecha. Para jugar debe empezar por tocar un cuadro del panel, no importa si es de la derecha o de la izquierda. Una vez lo haya tocado se le revelará una imagen si ha tocado un cuadro de la izquierda, o un texto acompañado de un audio si ha tocado un cuadro de la derecha (Véase imagen A.3). Si ha tocado un solo cuadro, todos los cuadros restantes de ese mismo lado se bloquearán hasta que toque un cuadro del lado contrario. En caso de que toque un cuadro de cada lado del panel, pueden ocurrir dos situaciones: la primera es que los cuadros no sean pareja, en cuyo caso, pasados unos segundos ambos cuadros volverán a ocultarse pudiendo seleccionar cualquiera de los cuadros de nuevo; la segunda situación se da si ambos cuadros son pareja, en cuyo caso se sumará un punto en el marcador y ambos cuadros quedarán mostrados y bloqueados hasta finalizar la partida (Véase imagen A.3). El número máximo de puntos es 12.

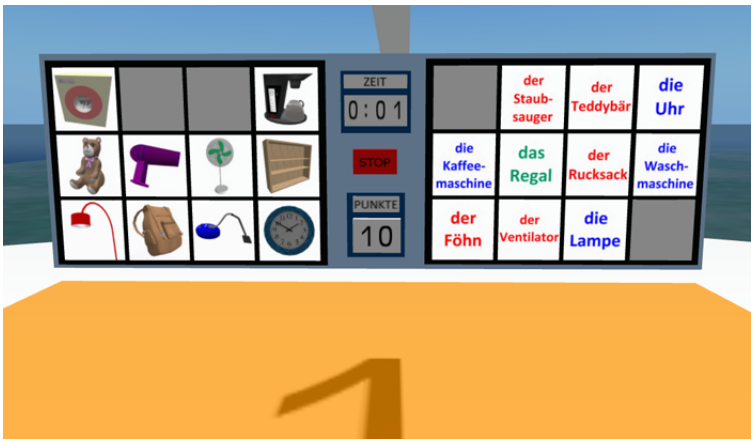


Figura A.3: Ejemplo de juego del panel 1.

Si al finalizar la partida el jugador ha obtenido en el panel una puntuación de 12 puntos, se emitirá un sonido de victoria. En caso de no haber conseguido 12 puntos se emitirá un sonido de derrota. Tanto si has obtenido 12 puntos como si has obtenido menos de 12 puntos, puedes volver a repetir el panel las veces que quieras.

A.3.2. Puntuación

Existe una pantalla con un botón azul, situada entre dos paneles y al lado de un cartel, que sirve para consultar la puntuación total de todos los paneles. Debes tocar el botón y consultar la pantalla. En la pantalla aparecerá el nombre del estudiante y la puntuación máxima obtenida en todos los paneles (Véase imagen A.4).

Figura A.4: Pantalla para consultar puntuación.

A.3.3. Teletranporte

En el centro de la sala existe un cúpula gris con un halo morado (Véase imagen A.5). Esta construcción es el teletransporte que te llevará a la actividad 2. Para usar la actividad 2 debes haber adquirido algunos conocimientos y cierta destreza en la actividad 1.

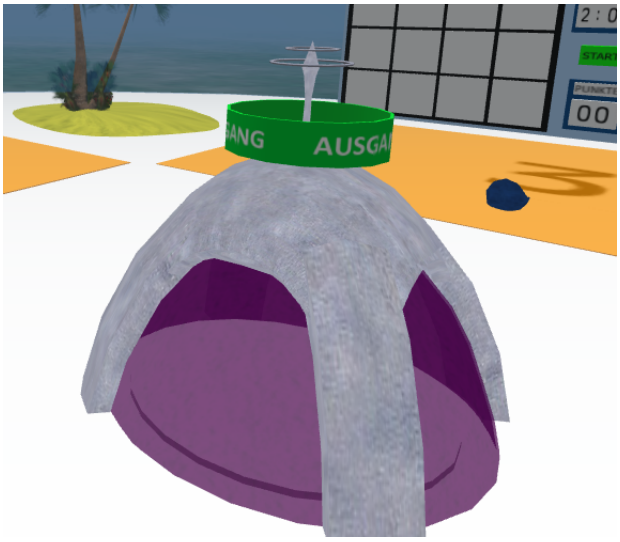


Figura A.5: Teletransporte de la actividad 1.

A.4. Actividad 2

La actividad esta formada por dos habitaciones contiguas. La primera habitación la encontramos al entrar por la puerta principal, que es el lugar donde nos deja el teletransporte de la actividad 1 (Véase imagen A.6). Para entrar a la segunda debemos atravesar la primera habitación, y al fondo a la izquierda nos encontramos una puerta con un cartel sobre ella que pone “Ausgang” (Véase imagen A.7, puerta de la derecha con cartel verde).

Figura A.6: Vista de la entrada a la primera habitación de la actividad 2.

Cruzamos por la puerta y podemos ver de izquierda a derecha, una fuente, una pantalla donde podremos consultar la puntuación de la actividad, y un pasillo de cristal y madera (Véase imagen A.7, a través de la ventana). A diferencia de la actividad 1, la actividad 2 solo se puede realizar de forma correcta en pareja. Cada miembro de la pareja debe tener un objeto distinto que se detallará a continuación.

Figura A.7: Vista de la entrada a la segunda habitación de la actividad 2.

A.4.1. Visor

Uno de los avatares debe vestirse el visor de imágenes. Para ello debemos acceder al *Inventario* y buscar el visor. Una vez encontrado debe seleccionarlo con el botón derecho del ratón y elegir la opción *vestir*. Una vez realizada esta acción con éxito, debería de aparecer en la esquina superior derecha (Véase imagen A.10).

El avatar que controle el visor de imágenes será el encargado de dar comienzo a la actividad, así como pararla cuando lo crean conveniente.

El visor contiene cuatro perspectivas de como debe quedar la casa al finalizar la actividad. Para seleccionar las diferentes imágenes debe tocar los botones con flechas (Véase imagen A.8). En caso de que la imagen del visor no se aprecie correctamente puede agrandarla tocando el centro de la imagen. Para devolver el visor a su estado normal debe volver a tocar el centro de la imagen(Véase imagen A.8).

Figura A.8: Vista del visor de imágenes.

Una vez haya acabado la actividad, para quitar el visor de la pantalla volvemos al *Inventario*, buscamos nuevamente el visor, y seleccionándolo con el botón derecho del ratón elegimos la opción desvestir.

A.4.2. Mando

Uno de los avatares debe vestirse el mando. Para ello debemos acceder al *Inventario* y buscar el mando. Una vez encontrado debe seleccionarlo con el botón derecho del ratón y elegir la opción *vestir*. Una vez realizada esta acción con éxito, debería de aparecer en la parte inferior central, tapando parte del avatar (Véase imagen A.11).

El avatar que controle el mando será el encargado de mover los objetos por la habitación, mientras la actividad haya empezado. Para mover un objeto primero debe seleccionarlo tocando el objeto. Cuando haya seleccionado un objeto debería ver como se ilumina una parte del objeto.

El mando contiene cuatro botones en la parte izquierda y dos en la parte derecha. Los botones de la izquierda mueven los objetos delante, detrás, a la derecha y a la izquierda (Véase imagen A.9). Los botones de la derecha mueven el objeto hacia arriba y hacia abajo (Véase imagen A.9). El mando solo puede mover los objetos en el interior de la casa, y en caso de que se vaya a salir de los límites de la casa, el objeto no responderá a moverse en esa dirección.

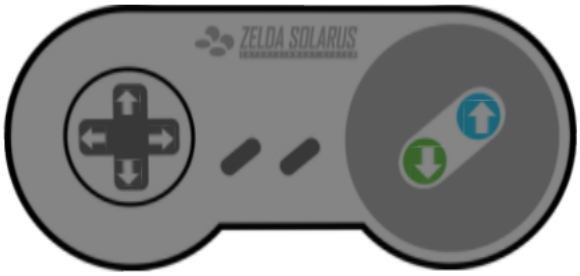


Figura A.9: Vista del mando.

Una vez haya acabado la actividad, para quitar el mando de la pantalla volvemos al *Inventario*, buscamos nuevamente el mando, y seleccionándolo con el botón derecho del ratón elegimos la opción desvestir.

A.4.3. Juego

Para iniciar el juego el avatar que lleva vestido el visor debe tocar el botón START. Al tocar START, verá que el tiempo comienza a contar y que el botón que antes era verde y ponía START en el centro, ahora ha cambiado, es rojo y pone STOP (Véase imagen A.10).

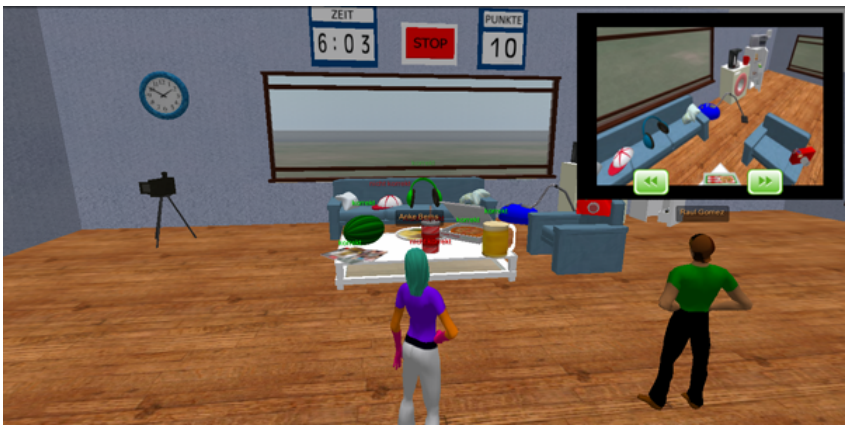


Figura A.10: Ejemplo de juego finalizado desde el punto de vista de un avatar con el visor.

El juego consiste en reorganizar los objetos situados en el interior de la primera habitación, usando para ello las imágenes del visor. Pero como hemos comentado anteriormente, el avatar que tiene el visor puede ver como quedaría la habitación ordenada pero no puede mover los objetos, y el avatar que tiene el mando puede mover los objetos pero no puede ver como quedaría la habitación ordenada. Para ordenar la habitación por tanto, deben comunicarse ambos estudiantes mediante el chat público (Véase imagen A.11).

Figura A.11: Ejemplo de juego finalizado desde el punto de vista de un avatar con el mando.

Para finalizar la actividad el avatar que lleva el visor puede pulsar el botón de STOP, o también se termina la actividad al finalizar el tiempo. Una vez finalizada la actividad se puede ver la puntuación obtenida en el marcador, y un texto encima de cada objeto con las palabras “korrekt” o “nicht korrekt”, dependiendo de si el objeto está en la posición correcta o no (Véase imagen A.11).

El número total de objetos, y por tanto de puntos, es 31. El tiempo máximo del que se dispone para completar la actividad son 8 minutos.

A.4.4. Puntuación

Existe una pantalla con un botón azul, situada en la habitación contigua a la habitación de la actividad, que sirve para consultar la puntuación total de la actividad. Debes tocar el botón y consultar la pantalla. En la pantalla aparecerá el nombre de cada componente de la pareja, la puntuación máxima obtenida en la actividad y el menor tiempo realizado en obtener la máxima puntuación (Véase imagen A.12).

Figura A.12: Pantalla que muestra la puntuación de la actividad 2.

A.4.5. Teletransporte

En la habitación contigua a la habitación de la actividad existe un pasillo de cristal y madera, indicado por una flecha como “Ausgang” (salida). Esta construcción es el teletransporte que te llevará a la actividad 1 (Véase imagen A.13).

Figura A.13: Teletransporte de la actividad 2.

Apéndice B

Manual de acceso al mundo virtual

Los pasos a seguir para poder acceder al mundo virtual son los siguientes:

B.1. Paso 1: Descargar e instalar un navegador

Existen varios tipos de navegadores con los que poder acceder al mundo virtual creado por *OpenSim*. Entre los más populares encontramos *Imprudence*, *RealXtend* o *Hippo*.

Descarga del navegador *Imprudence* [7]. Este ha sido el navegador utilizado para el desarrollo del proyecto. El navegador es fácil de manejar, posee un diseño sencillo e intuitivo, permite importar imágenes y sonidos fácilmente, y muy pocas veces se queda bloqueado respecto a los otros navegadores. Presenta un gran incoveniente en *Windows XP* con la reporducción de sonidos.
Descarga del navegador *RealXtend* [8]. Este navegador tiene características similares a *Imprudence*, aunque en ocasiones se ralentiza la imagen. Presenta algunos problemas al visualizar las texturas o reproducir los sonidos.

B.2. Paso 2: Conectarse al mundo virtual

Antes de poder acceder al mundo virtual es necesario registrarse en la página de VirtUAM [22] o bien usando los datos de nuestro servidor local. Para acceder al mundo virtual se puede seguir el siguiente video explicativo [16] o siguiendo los pasos que se detallan a continuación.

- **User Name**: Aquí el nombre del avatar completo, separado por un espacio entre nombre y apellido. Ejemplo “Cristina Utrera”.
- **Contraseña**: La contraseña introducida al registrarte.
- **Connect To**: En este campo se introduce la dirección del servidor seguido del puerto por el que se va a conectar. Como ejemplo se detalla la dirección de VirtUAM, “http://150.244.56.170:8002”.

Apéndice C

Manual de Instalación OpenSim

En este apéndice se muestran los requisítos mínimos, de hardware y software, que debe tener la máquina en la que se vaya a instalar el mundo virtual; como instalar el mundo virtual, y como arrancar el servidor del mundo virtual para ejecutarlo con los valores por defecto.

Es muy importante tener en cuenta que la versión que puede obtenerse desde el repositorio no posee todas las funcionalidades. Esta versión del juego esta preparada para almacenar datos en una base de datos externa a *OpenSim*, dentro de la plataforma VirtUAM, pero en el repositorio no se incluyen ni la base de datos, ni los archivos JSP de lectura y escritura en dicha base de datos. El teletransporte esta diseñado para que funcione en red, no de forma local.

C.1. Requisitos mínimos de hardware

Procesador: 1 Ghz (32 o 64 bits).

RAM: 2 GB para CPU de 32 bits y 3 GB para CPU de 64 bits.

Conexión a Internet: Mínimo 512 kb.

Espacio en disco: 1 GB.

Tarjeta gráfica: NVidia (GeForce 2, GeForce 4mx), ATI Radeon (8500, 9250 MB).

C.2. Requisitos mínimos software

C.2.1. SO Windows

OpenSimulator requiere *.NET Framework 3.5* cuando se ejecute en un entorno *Windows*. Tenga en cuenta que las versiones de *Windows* anteriores a NT o 2000 no son compatibles.
Si se ejecuta en *Windows* XP asegúrese de que esté actualizado, al menos el *Service Pack 2 (SP2)*.

C.2.2. SO Linux

OpenSimulator requiere *Mono 2.4.3* o posterior. [1]

CUIDADO:

[1] **Nota:** Para instalar *Mono* desde el repositorio ejecutar `sudo apt-get install mono-complete`

Se conoce que *OpenSimulator* tiene problemas de escalabilidad con las versiones de *Mono 2.8.x, 2.10.0 y 2.10.1*. A partir de *Mono 2.10.2*, los problemas de escalabilidad parecen haber sido resueltos.

Mono 2.6.x también parece estar bien, aunque la máquina virtual parece tener algunos problemas (problema con seguimiento de pila nativa) en simuladores que ejecutan muchas regiones o muchos usuarios/prims. Por lo tanto, debe usar *Mono 2.6.x , 2.10.2* o posterior. También puede usar *Mono 2.4.3*, pero es bastante antiguo.

C.2.3. Programa de control de versiones Subversion (o similar)

Windows

La descarga para el programa de control de versiones *TortoiseSVN*, se encuentra en el enlace [9].

Linux

La descarga del programa de control de versiones *Subversion* se puede hacer desde el repositorio:

```
sudo aptitude install subversion libapache2-svn
```

C.2.4. Base de datos

De forma predeterminada funciona con el motor *SQLite* pero haciendo los cambios pertinentes se pueden usar los motores *MySQL* o *MSSQL*.

La información referente a la configuración de estos motores se puede encontrar en el archivo OpenSim.ini.example y en otros archivos de ejemplo como *bin/config-include*. Si no desea utilizar la configuración por defecto de SQLite entonces debe configurar su base de datos.

- *SQLite* (por defecto) es una base de datos ligera que viene incluida con *OpenSimulator* y se puede utilizar sin necesidad de configuración adicional.
- *MySQL* es la base de datos recomendada para cualquier uso más allá de la experimentación o pequeñas aplicaciones independientes. La versión mínima de *MySQL* soportada es la 5.1.

  Sistemas Windows x64: En la actualidad existe un `bug_id= 5294` sin resolver. Se encuentra al ejecutar *OpenSimulator* con *MySQL 5.5* en sistemas *Windows* de 64 bits.
- *MSSQL*

C.3. Instalación en Windows

C.3.1. Instalación de OpenSim

Descargar con un programa de control de versiones (*Subversion*) el repositorio completo:

```
svn checkout http://aprendizaje-colaborativo-preposiciones-aleman-en-
opensim.googlecode.com/svn/trunk/ aprendizaje-colaborativo-preposicio
nes-aleman-en-opensim-read-only
```

Abrimos la consola y accedemos al directorio en el que se encuentra la carpeta que acabamos de descargar. Una vez dentro accedemos al directorio */OpenSim/bin*:

```
cd /<Ruta del repositorio>/OpenSim/bin
```

Ejecutamos `opensim.exe`:

```
opensim
```

C.4. Instalación en Linux

C.4.1. Instalación de OpenSim

Descargar con un programa de control de versiones (*Subversion*) el repositorio completo:

```
svn checkout http://aprendizaje-colaborativo-preposiciones-aleman-en-opensim.googlecode.com/svn/trunk/ aprendizaje-colaborativo-preposiciones-aleman-en-opensim-read-only
```

Abrimos la consola y accedemos al directorio en el que se encuentra la carpeta que acabamos de descargar. Una vez dentro accedemos al directorio */OpenSim/bin*:

```
cd /<Ruta del repositorio>/OpenSim/bin
```

Ejecutamos `mono opensim.exe`:

```
mono opensim
```

Bibliografía

[1] http://moodle.org.

[2] http://sakaiproject.org/.

[3] http://www.claroline.net/.

[4] http://www.blackboard.com.

[5] http://www.goingon.com/.

[6] http://www.edmodo.com/.

[7] Imprudence Viewer. http://imprudence.googlecode.com/files/Imprudence-1.3.2-Setup.exe.

[8] RealXtend Viewer. downloads.sourceforge.net/project/realxtendviewer/realxtendviewer/0.42/realxtendviewer_0.42.exe?r=&ts=1355007248&use_mirror=switch.

[9] TortoiseSVN. http://tortoisesvn.tigris.org/.

[10] Baran, B. Facebook as a formal instructional environment. *British Journal of Educational Technology*, page 41, 2010.

[11] Berns, A., Gonzalez-Pardo, A., and Camacho, D. Designing serious games for foreign language learning. *4th International Conference ICT for Language Learning, Florence (Italy)*, 2011.

[12] Berns, A., Gonzalez-Pardo, A., and Camacho, D. Implementing the use of virtual worlds in the teaching of foreign languages (level a1). *Proceedings of Learning a Language in Virtual Worlds: A Review of Innovation and ICT in Language Teaching Methodology*, pages 33 – 40, 2011.

[13] Berns, A., Gonzalez-Pardo, A., and Camacho, D. Game-like language learning in 3-d virtual environments. *Computers & Education*, 60(1):210–220, 2013.

[14] Garrison, D. and Kanuka, H. Blended learning: Uncovering its transformative potential in higher education. the internet and higher education. *The Internet and Higher Education*, 7:95–105, 2004.

[15] Gómez, R. Aprendizaje colaborativo de preposiciones en alemán en OpenSim. https://code.google.com/p/aprendizaje-colaborativo-preposiciones-aleman-en-opensim/.

[16] Gómez, R. Manual de acceso al mundo virtual. http://150.244.58.183:8080/PFCPrueba/Videos/TutorialAcceso_controller.swf.

[17] LindaKellie.com. http://www.lindakellie.com/.

[18] Madge, C., Meek, J., Wellens, J., and Hooley, T. Facebook, social integration and informal learning at university:it is more for socialising and talking to friends about work than for actually doing work. *Learning Media And Technology*, 34:141–155, 2009.

[19] Malone, T. What makes computer games fun? *Byte*, page 258 – 276, 1981.

[20] OpenSims Creations. http://opensim-creations.com/.

[21] Prensky, M. Digital game-based learning. *McGraw-Hill*, 2009.

[22] Universidad Autónoma de Madrid. VirtUAM, Virtual Worlds. http://aida.ii.uam.es/virtuam/.

GNU Free Documentation License

Version 1.3, 3 November 2008

<http://fsf.org/>

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.
A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.
A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.
The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.
A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".
Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.
The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.
The "**publisher**" means any person or entity that distributes copies of the Document to the public.
A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.
The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.
You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.
If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.
If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.
It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.
You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.
You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.
The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.
In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.
If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.
If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.
However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.
Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.
Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.
Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.
"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.
"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.
An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.
The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.
If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.